ELSEVIER

Advances in Water Resources 25 (2002) 1125–1146

Advances in Water Resources

www.elsevier.com/locate/advwatres

Numerical methods for incompressible viscous flow

Hans Petter Langtangen [a,b,*], Kent-Andre Mardal [a,b], Ragnar Winther [b,c]

[a] Department of Scientific Computing, Simula Research Laboratory, Oslo, Norway
[b] Department of Informatics, University of Oslo, Oslo, Norway
[c] Department of Mathematics, University of Oslo, Oslo, Norway

Received 15 October 2001; received in revised form 14 January 2002; accepted 27 May 2002

Abstract

We present an overview of the most common numerical solution strategies for the incompressible Navier–Stokes equations, including fully implicit formulations, artificial compressibility methods, penalty formulations, and operator splitting methods (pressure/velocity correction, projection methods). A unified framework that explains popular operator splitting methods as special cases of a fully implicit approach is also presented and can be used for constructing new and improved solution strategies. The exposition is mostly neutral to the spatial discretization technique, but we cover the need for staggered grids or mixed finite elements and outline some alternative stabilization techniques that allow using standard grids. Emphasis is put on showing the close relationship between (seemingly) different and competing solution approaches for incompressible viscous flow.

1. Introduction

Incompressible viscous flow phenomena arise in numerous disciplines in science and engineering. The simplest viscous flow problems involve just one fluid in the laminar regime. The governing equations consist in this case of the incompressible Navier–Stokes equations,

$$\frac{\partial \boldsymbol{v}}{\partial t} + \boldsymbol{v}\cdot\nabla\boldsymbol{v} = -\frac{1}{\varrho}\nabla p + \nu\nabla^2\boldsymbol{v} + \boldsymbol{g}, \tag{1}$$

and the equation of continuity, also called the incompressibility constraint,

$$\nabla\cdot\boldsymbol{v} = 0. \tag{2}$$

In these equations, $\boldsymbol{v}$ is the velocity field, p is the pressure, ϱ is the fluid density, $\boldsymbol{g}$ denotes body forces (such as gravity, centrifugal and Coriolis forces), ν is the kinematic viscosity of the fluid, and t denotes time. The initial conditions consist of prescribing $\boldsymbol{v}$, whereas the boundary conditions can be of several types: (i) prescribed velocity components, (ii) vanishing normal derivatives of velocity components, or (iii) prescribed stress vector components. The pressure is only determined up to a constant, but can be uniquely determined by prescribing the value (as a time series) at one spatial point. Many people refer to the system (1) and (2) as the Navier–Stokes equations. The authors will also adapt to this habit in the present paper.

Most flows in nature and technological devices are turbulent. The transition from laminar to turbulent flow is governed by the Reynolds number, $Re = Ud/\nu$, where U is a characteristic velocity of the flow and d is a characteristic length of the involved geometries. The basic Navier–Stokes equations describe both laminar and turbulent flow, but the spatial resolution required to resolve the small (and important) scales in turbulent flow makes direct solution of the Navier–Stokes equations too computationally demanding on today's computers. As an alternative, one can derive equations for the average flow and parameterize the effects of turbulence. Such common models for turbulent flow normally consist of two parts: one part modeling the average flow, and these equations are very similar to (1) and (2), and one part modeling the turbulent fluctuations. These two parts can at each time level be solved sequentially or in a fully coupled fashion. In the former case, one needs methods and software for the system (1) and (2) also in turbulent flow applications. Even in the fully coupled case the basic ideas regarding discretization of (1) and (2) are reused. We also mention that simulation of turbulence by solving the basic Navier–Stokes equations on very fine grids, referred to as direct numerical simulation (DNS), achieves increasing importance in turbulence

* Corresponding author.
E-mail address: hpl@ifi.uio.no (H.P. Langtangen).

0309-1708/02/$ - see front matter © 2002 Published by Elsevier Science Ltd.
PII: S0309-1708(02)00052-0

research as these solutions provide reference databases for fitting parameterized models.

In more complex physical flow phenomena, laminar or turbulent viscous flow is coupled with other processes, such as heat transfer, transport of pollution, and deformation of structures. Multi-phase/multi-component fluid flow models often involve equations of the type (1) and (2) for the total flow coupled with advection–diffusion-type equations for the concentrations of each phase or component. Many numerical strategies for complicated flow problems employ a splitting of the compound model, resulting in the need to solve (1) and (2) as one subset of equations in a possibly larger model involving lots of partial differential equations. Hence, it is evident that complex physical flow phenomena also demand software for solving (1) and (2) in a robust fashion.

Viscous flow models have important applications within the area of water resources. The common Darcy-type models for porous media flow are based on averaging viscous flow in a network of pores. However, the averaging introduces the permeability parameter, which must be measured experimentally, often with significant uncertainty. For multi-phase flow the ad hoc extensions of the permeability concept to relative permeabilities is insufficient for satisfactory modeling of many flow phenomena. Moreover, the extensions of Darcy's law to flow in fractured or highly porous media introduce considerable modeling uncertainty. A more fundamental approach to porous media flow is to simulate the viscous flow at the pore scale, in a series of network realizations, and compute the relation between the flow rate and the pressure differences. This is an important way to gain more insight into deriving better averaged flow models for practical use and to better understand the permeability concept [6,80]. The approach makes a demand for solving (1) and (2) in highly complex geometries, but the left-hand side of (1) can be neglected because of small Reynolds numbers (small characteristic length).

Water resources research and engineering are also concerned with free surface flow and currents in rivers, lakes, and the ocean. The commonly used models in these areas are based on averaging procedures in the vertical direction and ad hoc incorporation of viscous and turbulent effects. The shortcomings of averaged equations and primitive viscosity models are obvious in very shallow water, and in particular during run-up on beaches and inclined dam walls. Fully 3D viscous flow models based on (1) and (2) with free surfaces are now getting increased interest as these are becoming more accurate and computationally feasible [1,24,33,69,72,79].

Efficient and reliable numerical solution of the incompressible Navier–Stokes equations for industrial flow or water resources applications is extremely challenging. Very rapid changes in the velocity field may take place in thin boundary layers close to solid walls. Complex geometries can also lead to rapid local changes in the velocity. Locally refined grids, preferably in combination with error estimation and automatic grid adaption, are hence a key ingredient in robust methods. Most implicit solution methods for the Navier–Stokes equations end up with saddle-point problems, which complicates the construction of efficient iterative methods for solving the linear systems arising from the discretization process. Implicit solution methods also make a demand for solving large systems of nonlinear algebraic equations. Many incompressible viscous flow computations involve large-scale flow applications with several million grid points and thereby a need for the next generation of super-computers before becoming engineering or scientific practice. We have also mentioned that Navier–Stokes solvers are often embedded in much more complex flow models, which couple turbulence, heat transfer, and multi-specie fluids. Before attacking such complicated problems it is paramount that the numerical state-of-the-art of Navier–Stokes solvers is satisfactory. Turek [84] summarizes the results of benchmarks that were used to assess the quality of solution methods and software for unsteady flow around a cylinder in 2D and 3D. The discrepancy in results for the lifting force shows that more research is needed to develop sufficiently robust and reliable methods.

Numerical methods for incompressible viscous flow is a major part of the rapidly growing field *computational fluid dynamics* (CFD). CFD is now emerging as an operative tool in many parts of industry and science. However, CFD is not a mature field either from a natural scientist's or an application engineer's point of view; robust methods are still very much under development, many different numerical tracks are still competing, and reliable computations of complex multi-fluid flows are still (almost) beyond reach with today's methods and computers. We believe that at least a couple of decades of intensive research are needed to merge the seemingly different solution strategies and make them as robust as numerical models in, e.g., elasticity and heat conduction. Sound application of CFD today therefore requires advanced knowledge and skills both in numerical methods and fluid dynamics. To gain reliability in simulation results, it should be a part of common practice to compare the results from different discretizations, not only varying the grid spacings but also changing the discretization type and solution strategy. This requires a good overview and knowledge of different numerical techniques. Unfortunately, many CFD practitioners have a background from only one "numerical school" practicing a particular type of discretization technique and solution approach. One goal of the present paper is to provide a generic overview of the competing and most dominating methods in the part of CFD dealing with laminar incompressible viscous flow.

Writing a complete review of numerical methods for the Navier–Stokes equations is probably an impossible task. The book by Gresho and Sani [27] is a remarkable attempt to review the field, though with an emphasis on finite elements, but it required over 1000 pages and 48 pages of references. The page limits of a review paper demand the authors to only briefly report a few aspects of the field. Our focus is to present the basic ideas of the most fundamental solution techniques for the Navier–Stokes equations in a form that is accessible to a wide audience. The exposition is hence of the introductory and "engineering" type, keeping the amount of mathematical details to a modest level. We do not limit the scope to a particular spatial discretization technique, and therefore we can easily outline a common framework and reasoning which demonstrate the close connections between seemingly many different solution procedures in the literature. Hence, our hope is that this paper can help newcomers to the numerical viscous flow field see some structure in the jungle of Navier–Stokes solvers and papers, without having to start by digesting thick textbooks.

The literature on numerical solutions of the Navier–Stokes equations is overwhelming, and only a small fraction of the contributions is cited in this paper. Some books and reviews that the authors have found attractive are mentioned next. These references serve as good starting points for readers who want to study the contents of the present paper in more detail. Fletcher [21] contains a nicely written overview of some finite element and finite difference techniques for incompressible fluid flow (among many other topics). Gentle introductions to numerical methods and their applications to fluid flow can be found in the textbooks [3,20,28,58,59] (finite differences, finite volumes) and [60,65,66,89] (finite elements). More advanced texts include [15,25–27,30,61,84, 86]. Readers with a background in functional analysis and special interest in mathematics and finite element methods are encouraged to address Girault and Raviart [25] and the reviews by Dean and Glowinski [16] and Rannacher [63]. Readers interested in the efficiency of solution algorithms for the Navier–Stokes equations should consult Turek [84]. Gresho and Sani's comprehensive book [27] is accessible to a wide audience and contains thorough discussions of many topics that are weakly covered in most other literature, e.g., questions related to boundary conditions. The book's extensive report on practical experience with various methods is indispensable for CFD scientists, software developers, and consultants. An overview of CFD books is available on the Internet [36].

Section 2 describes the natural first approach to solving the Navier–Stokes equations and points out some basic numerical difficulties. Necessary conditions to ensure stable spatial discretizations are treated in Section 3. Thereafter we consider approximate solution strategies where the Navier–Stokes equations are transformed to more common and tractable systems of partial differential equations. These strategies include modern stabilization techniques (Section 4.1), penalty methods (Section 4.2), artificial compressibility (Section 4.3), and operator splitting techniques (Section 5). The latter family of strategies is popular and widespread and are known under many names in the literature, e.g., projection methods and pressure (or velocity) correction methods. We end the overview of operating splitting methods with a framework where such methods can be viewed as special preconditioners in an iterative scheme for a fully implicit formulation of the Navier–Stokes equations. Section 7 mentions some examples of existing software packages for solving incompressible viscous flow problems, and in Section 8 we point out important areas for future research.

2. A Naive derivation of schemes

With a background from a basic course in the numerical solution of partial differential equations, one would probably think of (1) as some kind of heat equation and try the simplest possible scheme in time, namely an explicit forward step

$$\frac{\boldsymbol{v}^{\ell+1}-\boldsymbol{v}^{\ell}}{\Delta t}+\boldsymbol{v}^{\ell}\cdot\nabla\boldsymbol{v}^{\ell}=-\frac{1}{\varrho}\nabla p^{\ell}+\nu\nabla^2\boldsymbol{v}^{\ell}+\boldsymbol{g}^{\ell}. \tag{3}$$

Here, Δt is the time step and superscript ℓ denotes the time level. The equation can be trivially solved for $\boldsymbol{v}^{\ell+1}$, after having introduced, e.g., finite elements [27], finite differences [3], finite volumes [20], or spectral methods [11] to discretize the spatial operators. However, the fundamental problem with this approach is that the new velocity $\boldsymbol{v}^{\ell+1}$ does not, in general, satisfy the other equation, i.e., $\nabla\cdot\boldsymbol{v}^{\ell+1}\neq 0$. Moreover, there is no natural computation of $p^{\ell+1}$.

A possible remedy is to introduce a pressure at $p^{\ell+1}$ in (3), which leaves two unknowns, $\boldsymbol{v}^{\ell+1}$ and $p^{\ell+1}$, and hence requires a simultaneous solution of

$$\boldsymbol{v}^{\ell+1}+\frac{\Delta t}{\varrho}\nabla p^{\ell+1}=\boldsymbol{v}^{\ell}-\Delta t\boldsymbol{v}^{\ell}\cdot\nabla\boldsymbol{v}^{\ell}+\Delta t\nu\nabla^2\boldsymbol{v}^{\ell}+\Delta t\boldsymbol{g}^{\ell}, \tag{4}$$

$$\nabla\cdot\boldsymbol{v}^{\ell+1}=0. \tag{5}$$

We can eliminate $\boldsymbol{v}^{\ell+1}$ by taking the divergence of (4) to obtain a Poisson equation for the pressure,

$$\nabla^2 p^{\ell+1}=\frac{\varrho}{\Delta t}\nabla\cdot(\boldsymbol{v}^{\ell}-\Delta t\boldsymbol{v}^{\ell}\cdot\nabla\boldsymbol{v}^{\ell}+\Delta t\nu\nabla^2\boldsymbol{v}^{\ell}+\Delta t\boldsymbol{g}^{\ell}). \tag{6}$$

However, there are no natural boundary conditions for $p^{\ell+1}$. Hence, solving (6) and then finding $\boldsymbol{v}^{\ell+1}$ trivially from (4) is therefore not in itself a sufficient solution strategy. More sophisticated variants of this method are considered in Section 5, but the lack of explicit boundary data for $p^{\ell+1}$ will remain a problem.

More implicitness of the velocity terms in (1) can easily be introduced. One can, for example, try a semi-implicit approach, based on a Backward Euler scheme, using an "old" velocity (as a linearization technique) in the convective term $\boldsymbol{v}\cdot\nabla\boldsymbol{v}$:

$$(1+\Delta t\boldsymbol{v}^{\ell}\cdot\nabla-\Delta t\nu\nabla^2)\boldsymbol{v}^{\ell+1}+\frac{\Delta t}{\varrho}\nabla p^{\ell+1} = \boldsymbol{v}^{\ell}+\Delta t\boldsymbol{g}^{\ell+1}, \tag{7}$$

$$\nabla\cdot\boldsymbol{v}^{\ell+1}=0. \tag{8}$$

This problem has the proper boundary conditions since (7) and (8) have the same order of the spatial operators as the original system (1) and (2). Using some discretization in space, one arrives in both cases at a linear system, which can be written on block form:

$$\begin{bmatrix}\boldsymbol{N} & \boldsymbol{Q}\\ \boldsymbol{Q}^{\mathrm{T}} & \boldsymbol{0}\end{bmatrix}\begin{bmatrix}\boldsymbol{u}\\ \boldsymbol{p}\end{bmatrix}=\begin{bmatrix}\boldsymbol{f}\\ \boldsymbol{0}\end{bmatrix}. \tag{9}$$

The vector $\boldsymbol{u}$ contains in this context all the spatial degrees of freedom (i.e., grid point values) of the vector field $\boldsymbol{v}^{\ell+1}$, whereas $\boldsymbol{p}$ is the vector of pressure degrees of freedom in the grid.

A fully implicit approach, using a backward Euler scheme for (1), where the convective term $\boldsymbol{v}\cdot\nabla\boldsymbol{v}$ is evaluated as $\boldsymbol{v}^{\ell+1}\cdot\nabla\boldsymbol{v}^{\ell+1}$, leads to a nonlinear equation in $\boldsymbol{v}^{\ell+1}$. Standard Newton or Picard iteration methods result in a sequence of matrix systems of the form at each time level.

In contrast to linear systems arising from standard discretization of, e.g., the diffusion equation, the system (9) may be singular. Special spatial discretization or stabilizing techniques are needed to ensure an invertible matrix in (9) and are reviewed in Sections 3 and 4. In the simplest case, $\boldsymbol{N}$ is a symmetric and positive definite matrix (this requires the convective term $\boldsymbol{v}\cdot\nabla\boldsymbol{v}$ to be evaluated explicitly at time level ℓ, such that the term appears on the right-hand side of (7)), and $\boldsymbol{Q}$ is a rectangular matrix. The stable spatial discretizations are designed such that the matrix $\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{Q}$ is non-singular. It should be noted that these conditions on $\boldsymbol{N}$ and $\boldsymbol{Q}$ lead to the property that the coefficient matrix in (9) is symmetric and non-singular but indefinite. This indefiniteness causes some difficulties. For example, a standard iterative method like the preconditioned conjugate gradient method cannot be directly used. In fact, preconditioners for these saddle-point problems are much more delicate to construct even when using more general solvers like, e.g., GMRES and may lead to breakdown if not constructed properly. Many of the time stepping procedures for the Navier–Stokes system have been partially motivated by the desire to avoid the solution of systems of the form (9). However, as we shall see later, such a strategy will introduce other difficulties.

3. Spatial discretization techniques

So far we have only been concerned with the details of the time discretization. Now we shall address spatial discretization techniques for the systems (4) and (5) or (7) and (8).

3.1. Finite differences and staggered grids

Initial attempts to solve the Navier–Stokes equations employed straightforward centered finite differences to the spatial operators on a regular grid, with the pressure and velocity components being unknown at the corners of each cell. Two typical terms in the equations would then be discretized as follows in a uniform 2D grid:

$$\left[\frac{\partial p}{\partial x}\right]_{i,j}^{\ell+1}\approx-\frac{p_{i+1,j}^{\ell+1}-p_{i-1,j}^{\ell+1}}{2\Delta x} \tag{10}$$

and

$$\left[\frac{\partial^2 u}{\partial y^2}\right]_{i,j}^{\ell}\approx\frac{u_{i,j-1}^{\ell}-2u_{i,j}^{\ell}+u_{i,j+1}^{\ell}}{\Delta y^2},$$

where Δx and Δy are uniform spatial cell sizes, $\phi_{i,j}^{\ell}$ means the numerical value of a function ϕ at the point with spatial index (i,j) at time level ℓ.

Two types of instabilities were soon discovered, associated with this type of spatial discretization. The pressure can be highly oscillatory or even undetermined by the discrete system, although the corresponding velocities may be well approximated. The reason for this phenomenon is that the symmetric difference operator (10) will annihilate checkerboard pressures, i.e., pressures which oscillate between 1 and -1 on each grid line connecting the grid points. In fact, if the vertices are colored in a checkerboard pattern, then the pressure at the black vertices will not be related to the pressure at the white vertices. Hence, the pressure is undetermined by the discrete system and wild oscillations or overflow will occur. This instability is related to whether the system (9) is singular or not. There is also a "softer" version of this phenomenon when (9) is nearly singular. Then the pressure will not necessarily oscillate, but it will not converge to the actual solution either.

The second type of instability is visible as non-physical oscillations in the velocities at high Reynolds numbers. This instability is the same as encountered when solving advection-dominated transport equations, hyperbolic conservation laws, or boundary layer equations and can be cured by well-known techniques, among which upwind differences represent the simplest approach. We shall not be concerned with this topic in the present paper, but the interested reader can consult the references [9,20,21,27,59,71] for effective numerical techniques.

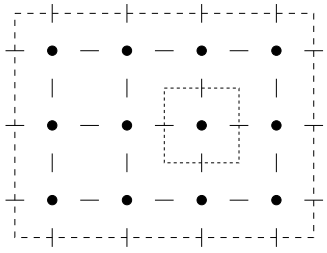

Fig. 1. Example on a staggered grid for the Navier–Stokes equations, where p and $\boldsymbol{v} = (u, v)$ are unknown at different spatial locations. The (●) denotes p points, (–) denotes u points, whereas (|) denotes v points.

The remedy for oscillatory or checkerboard pressure solutions is, in a finite difference context, to introduce a staggered grid in space. This means that the primary unknowns, the pressure and the velocity components, are sought at different points in the grid. Fig. 1 displays such a grid in 2D, and Fig. 2 zooms in on a cell and shows the spatial indices associated with the point values of the pressure and velocity components that enter the scheme.

Discretizing the terms $-\partial p/\partial x$ and $\partial^2 u/\partial y^2$ on the staggered grid at a point with spatial indices $(i, j+\frac{1}{2})$ now results in

$$-\left[\frac{\partial p}{\partial x}\right]^{\ell+1}_{i,j+\frac{1}{2}} \approx -\frac{p^{\ell+1}_{i+\frac{1}{2},j+\frac{1}{2}} - p^{\ell+1}_{i-\frac{1}{2},j+\frac{1}{2}}}{\Delta x}$$

and

$$\left[\frac{\partial^2 u}{\partial y^2}\right]^{\ell}_{i,j+\frac{1}{2}} \approx \frac{u^{\ell}_{i,j-\frac{1}{2}} - 2u^{\ell}_{i,j+\frac{1}{2}} + u^{\ell}_{i,j+\frac{3}{2}}}{\Delta y^2}.$$

The staggered grid is convenient for many of the derivatives appearing in the equations, but for the nonlinear terms it is necessary to introduce averaging. See, for instance, [3,20,21,28] for more details regarding discretization on staggered grids.

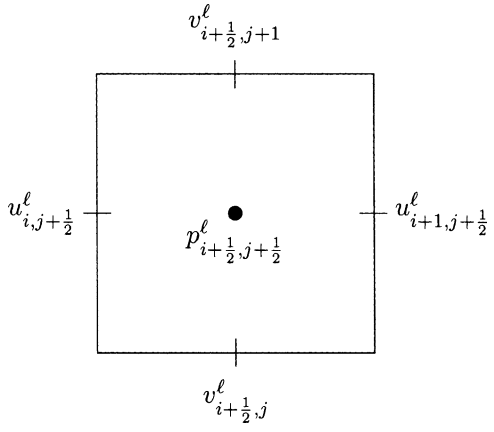

Fig. 2. A typical cell in a staggered grid.

Finite volume methods are particularly popular in CFD. Although the final discrete equations are similar to those obtained by the finite difference method, the reasoning is different. One works with the integral form of the Eqs. (1) and (2), obtained either by integrating (1) and (2) or by direct derivation from basic physical principles. The domain is then divided into control volumes. These control volumes are different for the integral form of (2) and the various components of the integral form of (1). For example, in the grid in Fig. 1 the dotted cell, also appearing in Fig. 2, is a typical control volume for the integral form of the equation of continuity, whereas the control volumes for the components of the equation of motion are shifted half a cell size in the various spatial directions. The governing equations in integral form involve volume/area integrals over the interior of a control volume and surface/line integrals over the sides. In the computation of these integrals, there is freedom to choose the type of interpolation for $\boldsymbol{v}$ and p and the numerical integration rules. Many CFD practitioners prefer finite volume methods because the derivation of the discrete equations is based directly on the underlying physical principles, thus resulting in "physically sound" schemes. From a mathematical point of view, finite volume, difference, and element methods are closely related, and it is difficult to decide that one approach is superior to the others; these spatial discretization methods have different advantages and disadvantages. We refer to textbooks like [3,20] for detailed information about the finite volume discretization technology.

Staggered grids are widespread in CFD. However, in recent years other ways of stabilizing the spatial discretization have emerged. Auxiliary terms in the equations or certain splittings of the operators in the Navier–Stokes equations can allow stable pressure solutions also on a standard grid. Avoiding staggered grids is particularly convenient when working with curvelinear grids. Much of the fundamental understanding of stabilizing the spatial discretization has arisen from finite element theory. We therefore postpone the discussion of particular stabilization techniques until we have reviewed the basics of finite element approximations to the spatial operators in the time-discrete Navier–Stokes equations.

3.2. Mixed finite elements

The staggered grids use different interpolation for the pressure and the velocity, hence we could call it mixed interpolation. In the finite element world the analog interpolation is referred to as mixed elements. The idea is, basically, to employ different basis functions for the different unknowns. A finite element function is expressed as a linear combination of a set of prescribed basis functions, also called shape functions or trial functions [88]. These basis functions are defined relative to a grid, which is a collection of elements (triangles,

qaudrilaterals, tetrahendra, or boxes), so the overall quality of a finite element approximation depends on the shape of the elements and the type of basis functions. Normally, the basis functions are lower-order polynomials over a single element.

One popular choice of basis functions for viscous flow is quadratic piecewise polynomials for the velocity components and linear piecewise polynomials for the pressure. This was in fact the spatial discretization used in the first report, by Taylor and Hood [76], on finite element methods for the Navier–Stokes equations.

The Babuska–Brezzi (BB) condition [8,25,27,30] is central for ensuring that the linear system of the form (9) is non-singular. Much of the mathematical theory and understanding of importance for the numerical solution of the Navier–Stokes equations has been developed for the simplified Stokes problem, where the acceleration terms on the left-hand side of (1) vanish:

$$\mathbf{0} = -\frac{1}{\varrho}\nabla p + \nu\nabla^2 \boldsymbol{v} + \boldsymbol{g}, \tag{11}$$

$$\nabla \cdot \boldsymbol{v} = 0. \tag{12}$$

We use the Galerkin method to formulate the discrete problem, seeking approximations

$$\boldsymbol{v} \approx \hat{\boldsymbol{v}} = \sum_{r=1}^{d}\sum_{i=1}^{n} v_i^r \boldsymbol{N}_i^r, \tag{13}$$

$$p \approx \hat{p} = \sum_{i=1}^{m} p_i L_i, \tag{14}$$

where $\boldsymbol{N}_i^r = N_i \boldsymbol{e}_r$, N_i and L_i are some scalar basis functions and $\boldsymbol{e}_r$ is the unity vector in the direction r. Here d is the number of spatial dimensions, i.e., 2 or 3. The number of velocity unknowns is dn, whereas the pressure is represented by m unknowns. Using N_i as weighting function for (11) and L_i as weighting function for (12), and integrating over Ω, one can derive a linear system for the coefficients v_i^r and p_i:

$$\sum_{j=1}^{n} \overline{N}_{ij} v_j^r + \sum_{j=1}^{m} Q_{ij}^r p_j = f_i^r, \quad i = 1,\ldots,dn,\ r = 1,\ldots,d, \tag{15}$$

$$\sum_{r=1}^{d}\sum_{j=1}^{n} Q_{ji}^r v_j^r = 0, \quad i = 1,\ldots,m, \tag{16}$$

where

$$\overline{N}_{ij} = \int_\Omega \nu \nabla N_i \cdot \nabla N_j \,\mathrm{d}\Omega, \tag{17}$$

$$\begin{aligned} Q_{ij}^r &= \frac{1}{\varrho}\int_\Omega \frac{\partial L_i}{\partial x_r} N_j \,\mathrm{d}\Omega \\ &= -\frac{1}{\varrho}\int_\Omega \frac{\partial N_i}{\partial x_r} L_j \,\mathrm{d}\Omega + \frac{1}{\varrho}\int_{\mathrm{d}\Omega} N_i L_j n_r \,\mathrm{d}\Gamma, \end{aligned} \tag{18}$$

$$f_i^r = \int_\Omega g^r \boldsymbol{N}_i^r \,\mathrm{d}\Omega. \tag{19}$$

We shall write such a system on block matrix form (like (9)):

$$\begin{bmatrix} \boldsymbol{N} & \boldsymbol{Q} \\ \boldsymbol{Q}^{\mathrm{T}} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{p} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{0} \end{bmatrix}, \quad \boldsymbol{N} = \begin{bmatrix} \overline{\boldsymbol{N}} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \overline{\boldsymbol{N}} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \overline{\boldsymbol{N}} \end{bmatrix},$$
$$\boldsymbol{Q} = \begin{bmatrix} \boldsymbol{Q}^1 \\ \boldsymbol{Q}^2 \\ \boldsymbol{Q}^3 \end{bmatrix}. \tag{20}$$

Here $\overline{\boldsymbol{N}}$ is the matrix with elements $\overline{N}_{ij}$, $\boldsymbol{Q}^r$ has elements Q_{ij}^r, and

$$\begin{aligned} \boldsymbol{u} &= (v_1^1,\ldots,v_n^1,v_1^2,\ldots,v_n^2,v_1^3,\ldots,v_n^3)^{\mathrm{T}}, \\ \boldsymbol{p} &= (p_1,\ldots,p_n)^{\mathrm{T}}. \end{aligned} \tag{21}$$

The $\boldsymbol{N}$ matrix is seen to be $dn \times dn$, whereas $\boldsymbol{Q}$ is $m \times dn$. In (20) and (21) we have assumed that $d = 3$. Moreover, we have multiplied the equation of continuity by the factor $-1/\varrho$ to obtain a symmetric linear system.

We shall now go through some algebra related to the block form of the Stokes problem, since this algebra will be needed later in Section 5.6. Let us write the discrete counterpart to (11) and (12) as

$$\boldsymbol{N}\boldsymbol{u} + \boldsymbol{Q}\boldsymbol{p} = \boldsymbol{f}\ , \tag{22}$$

$$\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{u} = \boldsymbol{0}. \tag{23}$$

The matrices $\boldsymbol{N}$ and $\boldsymbol{Q}$ can in principle arise from any spatial discretization method, e.g., finite differences, finite volumes, or finite elements, although we will specifically refer to the latter in what follows. First, we shall ask the question: What conditions on $\boldsymbol{N}$ and $\boldsymbol{Q}$ are needed to ensure that $\boldsymbol{u}$ and $\boldsymbol{p}$ are uniquely determined? We assume that $\boldsymbol{N}$ is positive definite (this assumption is actually the first part of the BB condition, which is satisfied for all "standard" elements). We can then multiply (22) by $\boldsymbol{N}^{-1}$ to obtain an expression for $\boldsymbol{u}$, which can be inserted in (23). The result is a linear system for $\boldsymbol{p}$:

$$-\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{N}^{-1}\boldsymbol{Q}\boldsymbol{p} = \boldsymbol{Q}^{\mathrm{T}}\boldsymbol{N}^{-1}\boldsymbol{f}\ . \tag{24}$$

Once the pressure is known, the velocities are found by solving

$$\boldsymbol{N}\boldsymbol{u} = (\boldsymbol{f} - \boldsymbol{Q}\boldsymbol{p}).$$

To obtain a uniquely determined $\boldsymbol{u}$ and $\boldsymbol{p}$, $\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{N}^{-1}\boldsymbol{Q}$, which is referred to as the *Schur complement*, must be non-singular. A necessary sufficient condition to ensure this is $\mathrm{Ker}(\boldsymbol{Q}) = \{0\}$, which is equivalent to requiring that

$$\sup_{\hat{\boldsymbol{v}}} \int_\Omega \hat{p}\nabla \cdot \hat{\boldsymbol{v}} > 0, \tag{25}$$

for all discrete pressure $\hat{p} \neq 0$, where the supremum is taken over all discrete velocities on the form (13). This guarantees solvability, but to get convergence of the numerical method, one also needs stability. Stability means in this setting that $\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{N}^{-1}\boldsymbol{Q}$ does not tend to a singular system as the h decrease. It is also sufficient to ensure optimal accuracy. This is where the famous BB condition comes in:

$$\inf_{p} \sup_{\hat{\boldsymbol{v}}} \frac{\int_{\Omega} \hat{p}\nabla \cdot \hat{\boldsymbol{v}}}{\|\hat{\boldsymbol{v}}\|_1 \|\hat{p}\|_0} \geqslant \gamma > 0. \tag{26}$$

Here, γ is independent of the discretization parameters, and the inf is taken over all $\hat{p} \neq 0$ on the form (14). The condition (26) is stated in numerous books and papers. Here we emphasize the usefulness of (26) as an operative tool for determining which elements for p and $\boldsymbol{v}$ that are "legal", in the sense that the elements lead to a solvable linear system and a stable, convergent method. For example, the popular choice of standard bilinear elements for $\boldsymbol{v}$ and piecewise constant elements for p violates (26), whereas standard quadratic triangular elements for $\boldsymbol{v}$ and standard linear triangules for p fulfill (26).

Provided the BB condition is fulfilled, with γ not depending on the mesh, one can derive an error estimate for the discretization of the Navier–Stokes equations:

$$\|\hat{\boldsymbol{v}} - \boldsymbol{v}\|_1 + \|\hat{p} - p\|_0 \leqslant C(h^k \|\boldsymbol{v}\|_{k+1} + h^{l+1}\|p\|_{l+1}), \tag{27}$$

This requires the exact solutions $\boldsymbol{v}$ and p to be in $[H^{k+1}(\Omega)]^d$ and $H^{l+1}(\Omega)$, respectively. The constant C is independent of the spatial mesh parameter h. The degree of the piecewise polynomial used for the velocity and the pressure is k and l, respectively, (see e.g. [29] or [25]). The inequality (27) involves the H^1 norm of $\boldsymbol{v}$, and the convergence rate of $\boldsymbol{v}$ in L^2 norm is one order higher. We see from the estimate (27) that $k = l + 1$ is the optimal choice, i.e., the velocity is approximated with accuracy of one higher order than the pressure. For example, the Taylor–Hood element [76] with quadratic velocity components and linear pressure gives quadratic and linear L_2-convergence in the mesh parameter for the velocities and pressure, respectively (under reasonable assumptions), see [5].

In simpler words, one could say that the computer resources are not wasted. We get what we *can* and *should* get. Elements that do not satisfy the BB condition may give an approximation that does not converge to the solution, and if it does, it may not converge as fast as one should expect from the order of the elements.

Numerous mixed finite elements satisfying the BB condition have been proposed over the years. However, elements not satisfying the BB condition may also work well. The element with bilinear velocities and constant pressure, which does violates the BB condition, is popular and usable in many occasions. A comprehensive review of mixed finite elements for incompressible viscous flow can be found in [27].

4. Stabilization techniques

Staggered grids or mixed finite elements can be challenging from an implementational point of view, especially when using unstructured, adaptive and/or hierarchical grids. Therefore, there has been significant interest in developing stabilization techniques which allow standard grids and equal order interpolation of $\boldsymbol{v}$ and p.

The singularity of the matrix (9) can be circumvented by introducing a stabilization matrix $\epsilon\boldsymbol{D}$ and possibly a perturbation of the right-hand side, $\epsilon\boldsymbol{d}$,

$$\begin{bmatrix} \boldsymbol{N} & \boldsymbol{Q} \\ \boldsymbol{Q}^{\mathrm{T}} & -\epsilon\boldsymbol{D} \end{bmatrix} \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{p} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f} \\ -\epsilon\boldsymbol{d} \end{bmatrix}, \tag{28}$$

where ϵ is a parameter that should be chosen either from physical knowledge or by other means. It can also be a spatially local parameter, which can be important for anisotropic meshes and boundary layer problems. There are mainly three methods used to construct $\epsilon\boldsymbol{D}$, all based on perturbed versions of the equation of continuity,

$$\nabla \cdot \boldsymbol{v} = \epsilon\nabla^2 p, \tag{29}$$

$$\nabla \cdot \boldsymbol{v} = -\epsilon p, \tag{30}$$

$$\nabla \cdot \boldsymbol{v} = -\epsilon \frac{\partial p}{\partial t}. \tag{31}$$

The approach (29) was derived with the purpose of stabilizing pressure oscillations and allowing standard grids and elements. Section 4.1 deals with this approach. The Eqs. (30) and (31) were not derived as stabilization methods, but were initiated from alternative physical and mathematical formulations of viscous incompressible flow, as we outline in Sections 4.2 and 4.3.

4.1. Pressure stabilization techniques

Finite elements not satisfying the BB condition often lead to non-physical oscillations in the pressure field. It may therefore be tempting to introduce a regularization based on $\nabla^2 p$, which will smooth the pressure solution [8]. One can show that the BB condition can be avoided by, e.g., introducing a stabilization term in the equation of continuity as shown in (29). It is common to write this perturbed equation with a slightly different perturbation parameter

$$\nabla \cdot \boldsymbol{v} = \epsilon h^2 \nabla^2 p, \tag{32}$$

where ϵ is a constant to be chosen. Now the velocities and the pressure can be represented with equal order, standard finite elements. We have introduced an $\mathcal{O}(h^2)$ perturbation of the problem, and there is hence no point in using higher-order elements. Consistent generalizations that also apply to higher-order elements have been proposed, a review can be found in Gresho and Sani [27] and Franca et al. in [30]. The idea behind these methods

is that one observes that by taking the divergence of (11) we get an equation that includes a $\nabla^2 p$ term like in (32),

$$\frac{1}{\varrho}\nabla^2 p = \nabla\cdot(\nu\nabla^2\boldsymbol{v}) + \nabla\cdot\boldsymbol{g}. \tag{33}$$

This divergence of (11) can be represented by the weak form

$$\int_\Omega \left(-\frac{1}{\varrho}\nabla\hat{p} + \nu\nabla^2\hat{\boldsymbol{v}} + \boldsymbol{g}\right)\cdot\nabla L_i \,\mathrm{d}\Omega = 0,$$

where the pressure basis functions are used as weighting functions. The left-hand side of this equation can then be added to (16) with a local weighting parameter ϵh_K^2 in each element. The result becomes

$$\sum_{r=1}^{d}\sum_{j=1}^{n}\widehat{Q}^r_{ji}v^r_j - \epsilon\sum_{j=1}^{m}D_{ij}p_j = -\epsilon d_i, \quad i=1,\ldots,m, \tag{34}$$

where

$$D_{ij} = \sum_K h_K^2\int_{\Omega_K}\nabla L_i\cdot\nabla L_j\,\mathrm{d}\Omega, \tag{35}$$

$$\widehat{Q}^r_{ij} = Q^r_{ij} + \epsilon\sum_K h_K^2\int_{\Omega_K}\nu\nabla^2 N^r_j\frac{\partial L_i}{\partial x_r}\,\mathrm{d}\Omega, \tag{36}$$

$$d^r_i = \sum_K h_K^2\int_{\Omega_K} g^r\frac{\partial L_i}{\partial x_r}\,\mathrm{d}\Omega. \tag{37}$$

The sum over K is to be taken over all elements; Ω_K is the domain of element K and h_K is the local mesh size. We see that this stabilization is not symmetric since $\widehat{Q}^r_{ij}\neq\widehat{Q}^r_{ji}$, however it is easy to see that a symmetric stabilization can be made by an adjustment of (15), such that

$$\int_\Omega\left(-\frac{1}{\varrho}\nabla\hat{p} + \nu\nabla^2\hat{\boldsymbol{v}} + \boldsymbol{g}\right)\cdot\nabla^2 N^r_i\,\mathrm{d}\Omega$$

is added to (15) with the same local weighting parameter. The use of second-order derivatives excludes linear polynomials for N^r_i. Detailed analysis of stabilization methods for both Stokes and Navier–Stokes equations can be found in [82].

One problem with stabilization techniques of the type outlined here is the choice of ϵ, since the value of ϵ influences the accuracy of the solution. If ϵ is too small we will experience pressure oscillations, and if ϵ is too large the accuracy of the solution deteriorates, since the solution is far from divergence free locally, although it is divergence free globally [27]. The determination of ϵ is therefore important. Several more or less complicated techniques exist, among the simplest is the construction of 'optimal bubbles' which is equivalent to the discretization using the MINI element [8,27]. Problems with this approach have been reported; one often experiences $\mathcal{O}(h)$ pressure oscillations in boundary layers with stretched elements, but a fix (multiply ϵ with a proper factor near the boundary layer) is suggested in [54]. An adaptive stabilization parameter calculated locally from properties of the element matrices and vectors is suggested in [78]. This approach gives a more robust method in the boundary layers.

4.2. Penalty methods

A well-known result from variational calculus is that minimizing a functional

$$J(v) = \int_\Omega |\nabla v|^2\,\mathrm{d}\Omega$$

over all functions v in the function space $H^1(\Omega)$, such that $v|_{\partial\Omega} = g$ where g is the prescribed boundary values, is equivalent to solving the Laplace problem

$$\nabla^2 u = 0 \text{ in } \Omega, \quad u = g \text{ on } \partial\Omega.$$

The Stokes problem (11) and (12) can be recast into a variational problem as follows: Minimize

$$J(\boldsymbol{w}) = \int_\Omega \varrho(\nu\nabla\boldsymbol{w}:\nabla\boldsymbol{w} - \boldsymbol{g}\cdot\boldsymbol{w})\,\mathrm{d}\Omega$$

over all $\boldsymbol{w}$ in some suitable function space, subject to the constraint

$$\nabla\cdot\boldsymbol{w} = 0.$$

Here, $\nabla\boldsymbol{w}{:}\nabla\boldsymbol{w} = \sum_r\sum_s w_{r,s}w_{r,s}$ is the "inner product" of two tensors (and $w_{r,s}$ means $\partial w_r/\partial x_s$). As boundary conditions, we assume that $\boldsymbol{w}$ is known or the stress vector vanishes, for the functional $J(\boldsymbol{w})$ to be correct (extension to more general conditions is a simple matter). This constrained minimization problem can be solved by the method of Lagrange multipliers: Find stationary points of

$$\widehat{J}(\boldsymbol{w},p) = J(\boldsymbol{w}) - \int_\Omega p\nabla\cdot\boldsymbol{w}\,\mathrm{d}\Omega$$

with respect to $\boldsymbol{w}$ and p, $-p$ being the Lagrange multiplier. The solution $(\boldsymbol{w},p)$ is a saddle point of $\widehat{J}$,

$$\widehat{J}(\boldsymbol{w},q)\leqslant\widehat{J}(\boldsymbol{w},p)\leqslant\widehat{J}(\boldsymbol{v},p)$$

and fulfills the Stokes problem (11) and (12).

The penalty method is a way of solving constrained variational problems approximately. One works with the modified functional

$$\widetilde{J}(\boldsymbol{w}) = J(\boldsymbol{w}) + \frac{1}{2}\lambda^2\int_\Omega(\nabla\cdot\boldsymbol{w})^2\,\mathrm{d}\Omega,$$

where λ is a prescribed, large parameter. The solution is governed by the equation

$$\frac{1}{\varrho}\lambda\nabla(\nabla\cdot\boldsymbol{v}) + \nu\nabla^2\boldsymbol{v} = \boldsymbol{g} \tag{38}$$

or the equivalent mixed formulation,

$$-\nu\nabla^2\boldsymbol{v} + \frac{1}{\varrho}\nabla p = \boldsymbol{g}, \tag{39}$$

$$\nabla \cdot \boldsymbol{v} + \frac{1}{\lambda} p = 0. \tag{40}$$

For numerical solution, (38) is a tremendous simplification at first sight; Eq. (38) is in fact equivalent to the equation of linear elasticity, for which robust numerical methods are well known. The penalty method does not seem to need mixed elements or staggered grids and is hence easy to implement.

The governing Eq. (38) is only an approximation to (11) and (12), where the latter model is obtained in the limit $\lambda \to \infty$. A too low λ leads to mass loss, whereas a large λ value leads to numerical difficulties (known as the locking problem in elasticity). Because of the large λ parameter, explicit time discretization leads to impractical small time steps, and implicit schemes in time are therefore used, with an associated demand of solving matrix systems. The disadvantage of the penalty method is that efficient *iterative* solution of these matrix systems is hard to construct. The discrete approximations of the system (38) will be positive definite. However, as λ approach infinity the system will tend to a discrete Stokes system, i.e., a discrete version of (39) and (40) with $1/\lambda = 0$. Hence, in the limit the elimination of the pressure is impossible, and this effect results in bad conditioning of the systems derived from (38) when λ is large. The dominating solution techniques have therefore been variants of Gaussian elimination. However, progress has been made with iterative solution techniques, see [67].

The penalty method has a firm theoretical basis for the Stokes problem [64]. Ad hoc extensions to the full Navier–Stokes equations are done by simply replacing Eq. (2) by

$$p = -\lambda \nabla \cdot \boldsymbol{v}$$

and eliminating the pressure p. This results in the governing flow equation

$$\frac{\partial \boldsymbol{v}}{\partial t} + \boldsymbol{v} \cdot \nabla \boldsymbol{v} = \frac{\lambda}{\varrho} \nabla (\nabla \cdot \boldsymbol{v}) + \nu \nabla^2 \boldsymbol{v} + \boldsymbol{g}. \tag{41}$$

In a sense, this is a nonlinear and time-dependent version of the standard linear elasticity equations.

One problem with the penalty method and standard elements is often referred to as *locking*. The locking phenomena can be illustrated by seeking a divergence-free velocity field subject to homogeneous Dirichlet boundary conditions on a regular finite element grid. For the standard linear elements the only solution to this problem is $\boldsymbol{v} = \boldsymbol{0}$. In the case of the penalty method we see that as $\lambda \to \infty$, $\boldsymbol{v} = \boldsymbol{0}$ is the only solution to (41) unless the matrix associated with the λ term is singular. One common way to avoid locking is, in a finite element context, to introduce *selective reduced integration*, which causes the matrix associated with the λ term to be singular. The selective reduced integration consists in applying a Gauss–Legendre rule to the λ term that is of one order lower than the rule applied to other terms (provided that rule is of minimum order for the problem in question). For example, if bilinear elements are employed for $\boldsymbol{v}$, the standard 2×2 Gauss–Legendre rule is used for all integrals, except those containing λ, which are treated by the 1×1 rule. The same technique is known from linear elasticity problems when the material approaches the incompressible limit. We refer to [34,64] or standard textbooks [65,66,89] for more details.

The use of selective reduced integration is justified by the fact that under certain conditions the reduced integration is equivalent to *consistent integration*, which is defined as the integration rule that is obtained if mixed elements were used to discretize (39) and (40) before eliminating the pressure to obtain (38). This equivalence result does, however, need some conditions on the elements. For instance, the difference between consistent and reduced integration was investigated in [19], and they reported much higher accuracy of mixed methods with consistent integration when using curved higher-order elements.

The locking phenomena is related to the finite element space and not to the equations themselves. For standard linear elements the incompressibility constraint will affect all degrees of freedom and therefore the approximation will be poor. Another way of circumventing this problem can therefore be to use elements where the incompressibility constraint will only affect some of the degrees of freedom, e.g., the element used to approximate Darcy–Stokes flow [55].

The penalty formulation can also be justified by physical considerations (Stokes' viscosity law [23]). We also mention that the method can be viewed as a *velocity* Schur complement method (cf. *pressure* Schur complement methods in Section 5.8). The augmented Lagrangian method is a regularization technique closely related to the penalty method. For a detailed discussion we refer to the book by Fortin and Glowinski [22].

4.3. Artificial compressibility methods

If there had been a term $\partial p/\partial t$ in the equation of continuity (2), the system of partial differential equation for viscous flow would be similar to the shallow water equations (with a viscous term). Simple explicit time stepping methods would then be applicable.

To introduce a pressure derivative in the equation of continuity, we consider the Navier–Stokes equations for compressible flow:

$$\frac{\partial \boldsymbol{v}}{\partial t} + \boldsymbol{v} \cdot \nabla \boldsymbol{v} = -\frac{1}{\varrho} \nabla p + \nu \nabla^2 \boldsymbol{v} + \boldsymbol{g}, \tag{42}$$

$$\frac{\partial \varrho}{\partial t} + \nabla \cdot (\varrho \boldsymbol{v}) = 0. \tag{43}$$

In (42) we have neglected the bulk viscosity since we aim at a model with small compressibility to be used as an

approximation to incompressible flow. The assumption of small compressibility, under isothermal conditions, suggests the linearized equation of state

$$p = p(\varrho) \approx p_0 + c_0^2(\varrho - \varrho_0), \tag{44}$$

where $c_0^2 = (\partial p/\partial \varrho)_0$ is the velocity of sound at the state (ϱ_0, p_0). We can now eliminate the density in the equation of continuity (43), resulting in

$$\frac{\partial p}{\partial t} + c_0^2 \varrho_0 \nabla \cdot \boldsymbol{v} = 0. \tag{45}$$

Eqs. (42) and (45) can be solved by, e.g., explicit forward differences in time. Here we list a second-order accurate leap–frog scheme, as originally suggested by Chorin [13]:

$$\frac{\boldsymbol{v}^{\ell+1} - \boldsymbol{v}^{\ell-1}}{2\Delta t} + \boldsymbol{v}^{\ell} \cdot \nabla \boldsymbol{v}^{\ell} = -\frac{1}{\varrho_0}\nabla p^{\ell} + \nu \nabla^2 \boldsymbol{v}^{\ell} + \boldsymbol{g}^{\ell}, \tag{46}$$

$$\frac{p^{\ell+1} - p^{\ell-1}}{2\Delta t} = -c_0^2 \varrho_0 \nabla \cdot \boldsymbol{v}^{\ell}. \tag{47}$$

This time scheme can be combined with centered spatial finite differences on standard grids or on staggered grids; Chorin [13] applied a DuFort–Frankel scheme on a standard grid. When solving the similar shallow water equations, most practitioners apply a staggered grid of the type in Fig. 1 as this give a more favorable numerical dispersion relation. Peyret and Taylor [59] recommend staggered grids for slightly compressible viscous flow for the same reason.

Artificial compressibility methods are often used to obtain a stationary solution. In this case, one can introduce $\alpha = \varrho_0 c_0^2$ and use α and Δt for optimizing a pseudo-time evolution of the flow towards a stationary state. A basic problem with the approach is that the time step Δt is limited by the inverse of c_0^2, which results in very small time steps when simulating incompressibility $(c_0 \to \infty)$. Implicit time stepping in (42) and (45) can then be much more efficient. In fact, explicit temporal schemes in (46) and (47) are closely related to operator splitting techniques (Sections 5 and 5.6), where the pressure Poisson equation is solved by a Jacobi-like iterative method [59]. Therefore, the scheme (46) and (47) is a very slow numerical method unless the flow exhibits rapid transient behavior of interest. Having said this, we should also add that artificial compressibility methods with explicit time integration have been very popular because of the trivial implementation and parallelization.

5. Operator splitting methods

The most popular numerical solution strategies today for the Navier–Stokes equations are based on operator splitting. This means that the system (1) and (2) is split into a series of simpler, familiar equations, such as advection equations, diffusion equations, advection–diffusion equations, Poisson equations, and explicit/implicit updates. Efficient numerical methods are much easier to construct for these standard equations than for the original system (1) and (2) directly. In particular, the evolution of the velocity consists of two main steps. First we neglect the incompressibility condition and compute a predicted velocity. Thereafter, the velocity is corrected by performing "a projection" onto the divergence free vector fields.

5.1. Explicit schemes

To illustrate the basics of operator splitting ideas, we start with a forward step in (1):

$$\boldsymbol{v}^{\ell+1} = \boldsymbol{v}^{\ell} - \Delta t \boldsymbol{v}^{\ell} \cdot \nabla \boldsymbol{v}^{\ell} - \frac{\Delta t}{\varrho}\nabla p^{\ell} + \Delta t \nu \nabla^2 \boldsymbol{v}^{\ell} + \Delta t \boldsymbol{g}^{\ell}. \tag{48}$$

The problem is that $\boldsymbol{v}^{\ell+1}$ does not satisfy the equation of continuity (2), i.e., $\nabla \cdot \boldsymbol{v}^{\ell+1} \neq 0$. Hence, we cannot claim that $\boldsymbol{v}^{\ell+1}$ in (48) is the velocity at the new time level $\ell + 1$. Instead, we view this velocity as a *predicted* (also called tentative or intermediate) velocity, denoted here by $\boldsymbol{v}^*$, and try to use the incompressibility constraint to compute a correction $\boldsymbol{v}^c$ such that $\boldsymbol{v}^{\ell+1} = \boldsymbol{v}^* + \boldsymbol{v}^c$. For more flexible control of the pressure information used in the equation for $\boldsymbol{v}^*$ we multiply the pressure term ∇p^{ℓ} by an adjustable factor β:

$$\boldsymbol{v}^* = \boldsymbol{v}^{\ell} - \Delta t \boldsymbol{v}^{\ell} \cdot \nabla \boldsymbol{v}^{\ell} - \Delta t \frac{\beta}{\varrho}\nabla p^{\ell} + \Delta t \nu \nabla^2 \boldsymbol{v}^{\ell} + \Delta t \boldsymbol{g}^{\ell}. \tag{49}$$

The $\boldsymbol{v}^{\ell+1}$ velocity to be sought should fulfill (48) with the pressure being evaluated at time level $\ell + 1$ (cf. Section 2):

$$\boldsymbol{v}^{\ell+1} = \boldsymbol{v}^{\ell} - \Delta t \boldsymbol{v}^{\ell} \cdot \nabla \boldsymbol{v}^{\ell} - \frac{\Delta t}{\varrho}\nabla p^{\ell+1} + \Delta t \nu \nabla^2 \boldsymbol{v}^{\ell} + \Delta t \boldsymbol{g}^{\ell}.$$

Subtracting this equation and the equation for $\boldsymbol{v}^*$ yields an expression for $\boldsymbol{v}^c$:

$$\boldsymbol{v}^c = \boldsymbol{v}^{\ell+1} - \boldsymbol{v}^* = -\frac{\Delta t}{\varrho}\nabla(p^{\ell+1} - \beta p^{\ell}).$$

That is,

$$\boldsymbol{v}^{\ell+1} = \boldsymbol{v}^* - \frac{\Delta t}{\varrho}\nabla(p^{\ell+1} - \beta p^{\ell}).$$

We must require $\nabla \cdot \boldsymbol{v}^{\ell+1} = 0$ and this leads to a Poisson equation for the pressure difference $\phi \equiv p^{\ell+1} - \beta p^{\ell}$:

$$\nabla^2 \phi = \frac{\varrho}{\Delta t}\nabla \cdot \boldsymbol{v}^*. \tag{50}$$

After having computed ϕ from this equation, we can update the pressure and the velocity:

$$p^{\ell+1} = \beta p^{\ell} + \phi, \tag{51}$$

$$\boldsymbol{v}^{\ell+1} = \boldsymbol{v}^* - \frac{\Delta t}{\varrho}\nabla \phi. \tag{52}$$

An open question is how to assign suitable boundary conditions to ϕ; the function, its normal derivative, or a

combination of the two must be known at the complete boundary since ϕ fulfills a Poisson equation. On the other hand, the pressure only needs to be specified (as a function of time) at a single point in space, when solving the original problem (1) and (2). There are two ways of obtaining the boundary conditions. One possibility is to compute $\partial p/\partial n$ from (1), just multiply by the unit normal vector at the boundary. From these expressions one can set up $\partial\phi/\partial n$. The second way of obtaining the boundary conditions is derived from (52); if $\boldsymbol{v}^{\ell+1}$ is supposed to fulfill the Dirichlet boundary conditions then

$$\nabla\phi|_{\partial\Omega} = \frac{\Delta t}{\varrho}(\boldsymbol{v}^{\ell+1} - \boldsymbol{v}^*)|_{\partial\Omega} = 0, \tag{53}$$

since $\boldsymbol{v}^*$ already has the proper boundary conditions. This relation is valid on all parts of the boundary where the velocity is prescribed. Because ϕ is the solution of (50), $\partial\phi/\partial n$ can be controlled, but these homogeneous boundary conditions are in conflict with the ones derived from (1) and (52) [61]. We see that the boundary conditions can be derived in different ways, and the surprising result is that one arrives at different conditions. Additionally we see that after the update (52) we are no longer in control of the tangential part of the velocity at the boundary. The problem with assigning proper boundary conditions for the pressure may result in a large error for the pressure near the boundary. Often one experiences an $\mathcal{O}(1)$ error in a boundary layer with width $\approx\sqrt{\Delta t\nu}$. This error can often be removed by extrapolating pressure values from the interior domain to the boundary. We refer to Gresho and Sani [27] for a thorough discussion of boundary conditions for the pressure Poisson equation.

The basic operator splitting algorithm can be summarized as follows.

1. Compute the prediction $\boldsymbol{v}^*$ from the explicit equation (49).
2. Compute ϕ from the Poisson equation (50).
3. Compute the new velocity $\boldsymbol{v}^{\ell+1}$ and pressure $p^{\ell+1}$ from the explicit equations (51) and (52).

Note that all steps are trivial numerical operations, except for the need to solve the Poisson equation, but this is a much simpler equation than the original problem (1) and (2).

5.2. Implicit velocity step

Operator splittings based on implicit difference schemes in time are more robust and stable than the explicit strategy just outlined. To illustrate how more implicit schemes can be constructed, we can take a backward step in (1) to obtain a predicted velocity $\boldsymbol{v}^*$:

$$\boldsymbol{v}^* + \Delta t\boldsymbol{v}^*\cdot\nabla\boldsymbol{v}^* + \Delta t\frac{\beta}{\varrho}\nabla p^{\ell} - \Delta t\nu\nabla^2\boldsymbol{v}^* + \Delta t\boldsymbol{g}^{\ell+1} = \boldsymbol{v}^{\ell}. \tag{54}$$

Alternatively, we could use the more flexible θ-rule in time (see below). Eq. (54) is nonlinear, and a simple linearization strategy is to use $\boldsymbol{v}^{\ell}\cdot\nabla\boldsymbol{v}^*$ instead of $\boldsymbol{v}^*\cdot\nabla\boldsymbol{v}^*$,

$$\boldsymbol{v}^* + \Delta t\boldsymbol{v}^{\ell}\cdot\nabla\boldsymbol{v}^* + \Delta t\frac{\beta}{\varrho}\nabla p^{\ell} - \Delta t\nu\nabla^2\boldsymbol{v}^* + \Delta t\boldsymbol{g}^{\ell+1} = \boldsymbol{v}^{\ell}. \tag{55}$$

Also in the case we keep the nonlinearity, most linearization methods end up with solving a sequence of convection–diffusion equations like (55). The $\boldsymbol{v}^{\ell+1}$ velocity is supposed to fulfill

$$\boldsymbol{v}^{\ell+1} + \Delta t\boldsymbol{v}^{\ell}\cdot\nabla\boldsymbol{v}^{\ell+1} + \frac{\Delta t}{\varrho}\nabla p^{\ell+1} - \Delta t\nu\nabla^2\boldsymbol{v}^{\ell+1} + \Delta t\boldsymbol{g}^{\ell+1} = \boldsymbol{v}^{\ell}.$$

The correction $\boldsymbol{v}^c$ is now $\boldsymbol{v}^{\ell+1} - \boldsymbol{v}^*$, i.e.,

$$\boldsymbol{v}^c = \boldsymbol{s}(\boldsymbol{v}^c) + \frac{\Delta t}{\varrho}\nabla\phi, \quad \boldsymbol{s}(\boldsymbol{v}^c) = \Delta t(-\boldsymbol{v}^{\ell}\cdot\nabla\boldsymbol{v}^c + \nu\nabla^2\boldsymbol{v}^c). \tag{56}$$

Note that so far we have not done anything "illegal", and this system can be written as a mixed system,

$$\boldsymbol{v}^c - \boldsymbol{s}(\boldsymbol{v}^c) + \frac{\Delta t}{\varrho}\nabla\phi = 0, \tag{57}$$

$$\nabla\cdot\boldsymbol{v}^c = \nabla\cdot\boldsymbol{v}^*. \tag{58}$$

It is then common to neglect or simplify $\boldsymbol{s}$, such that the problem changes into a mixed formulation of the Poisson equation,

$$\boldsymbol{v}^c - \frac{\Delta t}{\varrho}\nabla\phi = 0, \tag{59}$$

$$\nabla\cdot\boldsymbol{v}^c = \nabla\cdot\boldsymbol{v}^*. \tag{60}$$

Elimination of $\boldsymbol{v}^c$ yields a Poisson equation like (50),

$$\nabla^2\phi = \frac{\varrho}{\Delta t}\nabla\cdot\boldsymbol{v}^*. \tag{61}$$

The problems at the boundary that were discussed in the previous section apply to this method as well. Different choices of and approximations to $\boldsymbol{s}$ give rise to different methods. We shall come back to this point later when discretizing in space prior to splitting the original equations.

To summarize, the sketched implicit operator splitting method consists of solving an advection–diffusion equation (55), a Poisson equation (61), and then performing two explicit updates (we assume that $\boldsymbol{s}$ is neglected):

$$\boldsymbol{v}^{\ell+1} = \boldsymbol{v}^* - \nabla\frac{\Delta t}{\varrho}\phi, \tag{62}$$

$$p^{\ell+1} = \beta p^{\ell} + \phi. \tag{63}$$

The outlined operator splitting approaches reduce the Navier–Stokes equations to a system of standard equations (explicit updates, linear convection–diffusion equations, and Poisson equations). These equations can be discretized by standard finite elements, that is, there is seemingly no need for mixed finite elements, a fact that simplifies the implementation of viscous flow simulators significantly.

5.3. A more accurate projection method

In Brown et al. [10] an attempt to remove the boundary layer introduced in the pressure by the projection method discussed above is described, cf. also [17,52]. Previously, in Sections 5.1 and 5.2, we neglected the term $\boldsymbol{s}(\boldsymbol{v}^c)$, since the scheme was only first-order in time. This resulted in a problem with the boundary conditions on ϕ. If the scheme is second-order in time, we cannot remove this term. In [10] a second-order scheme in *velocity and pressure* is described. In addition, many previous attempts to construct second-order methods for the incompressible Navier–Stokes equations are reviewed there. In order to describe the approach let us start to form a centered scheme at time level $\ell+1/2$ for the momentum equation:

$$\frac{\boldsymbol{v}^{\ell+1}-\boldsymbol{v}^{\ell}}{\Delta t}+\nabla p^{\ell+1/2}=-[\boldsymbol{v}\cdot\nabla\boldsymbol{v}]^{\ell+1/2}+\frac{\nu}{2}\nabla^2(\boldsymbol{v}^{\ell+1}+\boldsymbol{v}^{\ell})+\boldsymbol{g}^{\ell+1/2}, \tag{64}$$

$$\nabla\cdot\boldsymbol{v}^{\ell+1}=0. \tag{65}$$

Here the approximation $[\boldsymbol{v}\cdot\nabla\boldsymbol{v}]^{\ell+1/2}$ is assumed to be extrapolated from the solution on previous time levels. A predicted velocity is computed by

$$\frac{\boldsymbol{v}^{*,\ell+1}-\boldsymbol{v}^{*,\ell}}{\Delta t}=-[\boldsymbol{v}\cdot\nabla\boldsymbol{v}]^{\ell+1/2}+\frac{\nu}{2}\nabla^2(\boldsymbol{v}^{*,\ell+1}+\boldsymbol{v}^{*,\ell})+\boldsymbol{g}^{\ell+1/2}. \tag{66}$$

Note that since we now use $\boldsymbol{v}^{*,\ell}$ instead of $\boldsymbol{v}^{\ell}$ as the initial solution at level ℓ, $\boldsymbol{v}^*$ follows its own evolution equation. The initial conditions $\boldsymbol{v}^{*,0}=\boldsymbol{v}^0$ should be used. Subtracting (66) from (64) we obtain an equation for the velocity correction, $\boldsymbol{v}^{c,\ell+1}=\boldsymbol{v}^{\ell+1}-\boldsymbol{v}^{*,\ell+1}$,

$$\frac{\boldsymbol{v}^{c,\ell+1}-\boldsymbol{v}^{c,\ell}}{\Delta t}+\nabla p^{\ell+1/2}=\frac{\nu}{2}\nabla^2(\boldsymbol{v}^{c,\ell+1}+\boldsymbol{v}^{c,\ell}). \tag{67}$$

Observe that this is a diffusion equation for $\boldsymbol{v}^c$ with a gradient, $-\nabla p$, as a forcing term. If we assume that this implies that $\boldsymbol{v}^c$ is itself a gradient we can conclude that

$$\boldsymbol{v}^{\ell+1}-\boldsymbol{v}^{*,\ell+1}=\boldsymbol{v}^{c,\ell+1}=\nabla\phi^{\ell+1} \tag{68}$$

for a suitable function $\phi^{\ell+1}$. From $\nabla\cdot\boldsymbol{v}^{\ell+1}=0$ we get

$$-\nabla^2\phi^{\ell+1}=-\nabla\cdot\boldsymbol{v}^{*,\ell+1}. \tag{69}$$

To solve this equation, it remains to assign proper boundary conditions to $\phi^{\ell+1}$. From the discussion in Section 5.1 we know that the boundary conditions on ϕ can be determined such that $\boldsymbol{v}^{l+1}$ fulfills the normal components (or one tangential component), i.e.,

$$\left.\frac{\partial\phi}{\partial n}\right|_{\partial\Omega}=\boldsymbol{n}\cdot(\boldsymbol{v}^{*,\ell+1}-\boldsymbol{v}^{l+1})|_{\partial\Omega}=0. \tag{70}$$

We have now fixed the normal components of the boundary conditions on $\boldsymbol{v}^{*,\ell+1}$, but we have lost control over the tangential part. In [10] they therefore propose to use an extrapolated value for $\hat{\boldsymbol{\phi}}^{\ell+1}$ to determined the tangential parts of $\boldsymbol{v}^*$ such that

$$\boldsymbol{t}\cdot\boldsymbol{v}^{*,\ell+1}|_{\partial\Omega}=\boldsymbol{t}\cdot(\boldsymbol{v}^{\ell+1}+\nabla\hat{\boldsymbol{\phi}}^{\ell+1})|_{\partial\Omega}, \tag{71}$$

where $\boldsymbol{t}$ is a tangent vector.

A relation between p and ϕ is computed by inserting (68) into (67) to get the pressure update,

$$p^{\ell+1}=\frac{\phi^{\ell+1}-\phi^{\ell}}{\Delta t}-\frac{\nu}{2}\nabla^2(\phi^{\ell+1}+\phi^{\ell}). \tag{72}$$

To summerize this approach a complete time step consists of

1. Evolve $\boldsymbol{v}^*$ by (66) and the boundary conditions given by (70) and (71).
2. Solve (69) for $\phi^{\ell+1}$ using the boundary condition (70).
3. Compute $\boldsymbol{v}^{\ell+1}$ and $p^{\ell+1}$ using (68) and (72).

We refer to Brown et al. [10] for more details. A critical and non-obivious step seems to be the correctness of the derivation of (68) from (67). This may depend on the given boundary conditions.

5.4. Relation to stabilization techniques

The operator splitting techniques in time, as explained in Sections 5.1 and 5.2, seem to work quite well in spite of their simplicity compared to the original coupled system (1) and (2). Some explanation of why the method works can be found in [62,63,73,74]. The point is that one can show that the operator splitting method from Section 5.2 is equivalent to solving a system like (1) and (2) with an old pressure in (1) and a stabilization term $\Delta t\nabla^2 p$ on the right-hand side of (2). This stabilization term makes it possible to use standard elements and grids. Other suggested operator splitting methods [63] can be interpreted as a $\Delta t\partial p/\partial t$ stabilization term in the equation of continuity, i.e., a method closely related to the artificial compressibility scheme from Section 4.3.

5.5. Fractional step methods

Fractional step methods constitute another class of popular strategies for splitting the Navier–Stokes equations. A typical fractional step approach [2,4,10,20,87] may start with a time discretization where the convective term is treated explicitly, whereas the pressure and the viscosity term are treated implicitly:

$$\boldsymbol{v}^{\ell+1} - \boldsymbol{v}^{\ell} + \Delta t \boldsymbol{v}^{\ell} \cdot \nabla \boldsymbol{v}^{\ell} = -\Delta t \frac{\beta}{\varrho} \nabla p^{\ell+1} + \Delta t \nu \nabla^2 \boldsymbol{v}^{\ell+1} + \Delta t \boldsymbol{g}^{\ell+1}, \tag{73}$$

$$\nabla \cdot \boldsymbol{v}^{\ell+1} = 0. \tag{74}$$

One possible splitting of (73) and (74) is now

$$\boldsymbol{v}^{*} - \boldsymbol{v}^{\ell} + \Delta t \boldsymbol{v}^{\ell} \cdot \nabla \boldsymbol{v}^{\ell} = 0, \tag{75}$$

$$\boldsymbol{v}^{**} = \boldsymbol{v}^{*} + \Delta t \nu \nabla^2 \boldsymbol{v}^{**} + \Delta t \boldsymbol{g}^{\ell+1}, \tag{76}$$

$$\boldsymbol{v}^{\ell+1} = \boldsymbol{v}^{**} - \frac{\Delta t}{\varrho} \nabla p^{\ell+1}, \tag{77}$$

$$\nabla \cdot \boldsymbol{v}^{\ell+1} = 0. \tag{78}$$

Notice that combining (75)–(77) yields (73). Eq. (75) is a pure advection equation and can be solved by appropriate explicit methods for hyperbolic problems. Eq. (76) is a standard heat conduction equation, with implicit time differencing. Finally, (77) and (78) is a mixed Poisson problem, which can be solved by special methods for mixed Poisson problems, or one can insert $\boldsymbol{v}^{\ell+1}$ from (77) into (78) to obtain a pressure Poisson equation,

$$\nabla^2 p^{\ell+1} = \frac{\varrho}{\Delta t} \nabla \cdot \boldsymbol{v}^{**}. \tag{79}$$

After having solved this equation for $p^{\ell+1}$, (77) is used to find the velocity $\boldsymbol{v}^{\ell+1}$ at the new time level. Using (79) and then (77) instead of solving (77) and (78) simultaneously has the advantage of avoiding staggered grids or mixed finite elements. However, (79) requires extra pressure boundary conditions at the whole boundary as discussed previously.

The fractional step methods offer flexibility in the splitting of the Navier–Stokes equations into equations that are significantly simpler to work with. For example, in the presented scheme, one can apply specialized methods to treat the $\boldsymbol{v} \cdot \nabla \boldsymbol{v}$ term because this term is now isolated in a Burgers equation (75) for which numerous accurate and efficient explicit solution methods exist. The implicit time stepping in the scheme is isolated in a standard heat or diffusion equation (76) whose solution can be obtained very efficiently. The last equation (79) is also a simple equation with a wealth of efficient solution methods. Although each of the equation can be solved with good control of efficiency, stability, and accuracy, it is an open question of how well the overall, compound solution algorithm behaves. *This is the downside of all operator splitting methods, and therefore these methods must be used with care.*

More accurate (second-order in Δt) fractional step schemes than outlined here can be constructed, see [16] for a framework and [10] for review.

5.6. Discretizing in space prior to discretizing in time

The numerical strategies in Sections 5.1–5.4 are based on discretizing (1) and (2) first in time, to get a set of simpler partial differential equations, and then discretizing the time-discrete equations in space. One fundamental difficulty with this approach is that we derive a second-order Poisson equation for the pressure itself or a pressure increment. Such a Poisson equation implies a demand for more boundary conditions for p than what is required in the original system (1) and (2), as discussed in the previous section. The cause of these problematic, and unnatural, boundary conditions on the pressure is the simplification of the system (57) and (58) to (59) and (60), where the term $\boldsymbol{s}(\boldsymbol{v}^c)$, containing $\Delta t \nu \nabla^2 \boldsymbol{v}^c$, is neglected. If we keep this term the system (57) and (58) is replaced by

$$(1 - \Delta t \nu \nabla^2)\boldsymbol{v}^c - \frac{\Delta t}{\varrho} \nabla \phi = 0, \tag{80}$$

$$\nabla \cdot \boldsymbol{v}^c = \nabla \cdot \boldsymbol{v}^{*}. \tag{81}$$

This system is a modified stationary Stokes system, which can be solved under the correct boundary conditions on the velocity field $\boldsymbol{v}^c$. However, this system cannot easily be reduced to a simple Poisson equation for the pressure increment ϕ. Instead, we have to solve the complete coupled system in $\boldsymbol{v}^c$ and ϕ, and when this system is discretized we obtain algebraic systems of the form (9). Hence, the implementation of the correct boundary conditions seems to be closely tied to the need to solve discrete saddle-point systems of the form (9).

Another attempt to avoid constructing extra consistent boundary conditions for the pressure is to first discretize the original system (1) and (2) in space. Hence, we need to discretize both the dynamic equation and the incompressibility conditions, using discrete approximations of the pressure and the velocity. This will lead to a system of ordinary differential equations with respect to time, with a set of algebraic constraints representing the incompressibility conditions, and with the proper boundary conditions built into the spatial discretization. A time stepping approach, closely related to operator splitting, for such constrained systems is to first facilitate an advancement of the velocity just using the dynamic equation. As a second step we then "project" the velocity onto the space of divergence free velocities. The two steps in this procedure are closely related to the approach discussed in Sections 5.1 and 5.2. For example, Eq. (55) can be seen as a dynamic step, while (59) and (60), or simply (61), can be seen as the projection step. However, the projection induced by the system (59) and (60) is not compatible with the boundary conditions of the original system (and this may lead to large error in the pressure near the boundary). In contrast, the

projection introduced by the system (80) and (81) has the correct boundary conditions.

In order to discuss this approach in greater detail let us apply either a finite element, finite volume, finite difference, or spectral method to discretize the spatial operators in the system (1) and (2). This yields a system of ordinary differential equations, which can be expressed in the following form:

$$\boldsymbol{M}\dot{\boldsymbol{u}} + \boldsymbol{K}(\boldsymbol{u})\boldsymbol{u} = -\boldsymbol{Q}\boldsymbol{p} + \boldsymbol{A}\boldsymbol{u} + \boldsymbol{f}\ , \tag{82}$$

$$\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{u} = \boldsymbol{0}. \tag{83}$$

Here, $\boldsymbol{u}$ is a vector of velocities at the (velocity) grid points, $\boldsymbol{p}$ is a vector of pressure values at the (pressure) grid points, $\boldsymbol{K}$ is a matrix arising from discretizing $\boldsymbol{v}\cdot\nabla$, $\boldsymbol{M}$ is a mass matrix (the identity matrix $\boldsymbol{I}$ in finite difference/volume methods), $\boldsymbol{Q}$ is a discrete gradient operator, $\boldsymbol{Q}^{\mathrm{T}}$ is the transpose of $\boldsymbol{Q}$, representing a discrete divergence operator, and $\boldsymbol{A}$ is a discrete Laplace operator. The right-hand side $\boldsymbol{f}$ contains body forces. Stable discretizations require mixed finite elements or staggered grids for finite volume and difference methods. Alternatively, one can add stabilization terms to the equations. The extra terms to be added to (82) and (83) are commented upon in Section 5.8.

We can easily devise a simple explicit method for (82) by using the same ideas as in Section 5.1. A tentative or predicted discrete velocity field $\boldsymbol{u}^*$ is computed by

$$\boldsymbol{M}\boldsymbol{u}^* = \boldsymbol{M}\boldsymbol{v}^\ell + \Delta t(-\boldsymbol{K}(\boldsymbol{u}^\ell)\boldsymbol{u}^\ell - \beta\boldsymbol{Q}\boldsymbol{p}^\ell + \boldsymbol{A}\boldsymbol{u}^\ell + \boldsymbol{f}^{\ \ell}). \tag{84}$$

A correction $\boldsymbol{u}^c$ is sought such that $\boldsymbol{u}^{\ell+1} = \boldsymbol{u}^* + \boldsymbol{u}^c$ fulfills $\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{u}^{\ell+1} = \boldsymbol{0}$. Subtracting $\boldsymbol{u}^*$ from $\boldsymbol{u}^{\ell+1}$ yields

$$\boldsymbol{u}^c = -\Delta t\boldsymbol{M}^{-1}\boldsymbol{Q}\boldsymbol{\phi}, \quad \boldsymbol{\phi} \equiv \boldsymbol{p}^{\ell+1} - \beta\boldsymbol{p}^\ell. \tag{85}$$

Now a projection step onto the constraint $\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{u}^{\ell+1} = \boldsymbol{0}$ results in an equation for $\boldsymbol{\phi}$:

$$\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{M}^{-1}\boldsymbol{Q}\boldsymbol{\phi} = \frac{1}{\Delta t}\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{u}^*. \tag{86}$$

This is a *discrete Poisson equation* for the pressure. For example, employing finite difference methods in a spatial staggered grid yields $\boldsymbol{M} = \boldsymbol{I}$ and $\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{Q}$ is then the standard 5- or 7-star discrete Laplace operator. The matrix $\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{M}^{-1}\boldsymbol{Q}$ is a counterpart to matrices arising from ∇^2 in the Poisson equations for ϕ in Sections 5.1 and 5.2.

Having computed $\boldsymbol{\phi}$, the new pressure and velocity values are found from

$$\boldsymbol{p}^{\ell+1} = \beta\boldsymbol{p}^\ell + \boldsymbol{\phi}, \tag{87}$$

$$\boldsymbol{u}^{\ell+1} = \boldsymbol{u}^* - \Delta t\boldsymbol{M}^{-1}\boldsymbol{Q}\boldsymbol{\phi}. \tag{88}$$

5.7. Classical schemes

In this subsection we shall present a common setting for many popular classical schemes for solving the Navier–Stokes equations. We start with formulating an implicit scheme for (82) using the θ-rule for flexibility; $\theta = 1$ gives the backward Euler scheme, $\theta = 1/2$ results in the trapezoidal rule (or a Crank–Nicolson scheme), and $\theta = 0$ recovers the explicit forward Euler scheme treated above. The time-discrete equations can be written as

$$\boldsymbol{N}\boldsymbol{u}^{\ell+1} + \Delta t\boldsymbol{Q}\boldsymbol{p}^{\ell+1} = \boldsymbol{q}, \tag{89}$$

$$\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{u}^{\ell+1} = \boldsymbol{0}, \tag{90}$$

where

$$\boldsymbol{N} = \boldsymbol{M} + \theta\Delta t\boldsymbol{R}(\boldsymbol{u}^\ell), \tag{91}$$

$$\boldsymbol{R}(\boldsymbol{u}^\ell) = \boldsymbol{K}(\boldsymbol{u}^\ell) - \boldsymbol{A}, \tag{92}$$

$$\boldsymbol{q} = (\boldsymbol{M} - (1-\theta)\Delta t\boldsymbol{R}(\boldsymbol{u}^\ell))\boldsymbol{u}^\ell + \Delta t\boldsymbol{f}^{\ \ell+1}) \tag{93}$$

are introduced to save space in the equations. Observe that we have linearized the convective term by using $\boldsymbol{R}(\boldsymbol{u}^\ell)$ on the left-hand side of (89). One could, of course, resolve the nonlinearity by some kind of iteration instead.

To proceed, we skip the pressure or use old pressure values in to produce a predicted velocity $\boldsymbol{u}^*$:

$$\boldsymbol{N}\boldsymbol{u}^* = \boldsymbol{q} - \beta\Delta t\boldsymbol{Q}\boldsymbol{p}^\ell. \tag{94}$$

The correction $\boldsymbol{u}^c = \boldsymbol{u}^{\ell+1} - \boldsymbol{u}^*$ is now governed by

$$\boldsymbol{N}\boldsymbol{u}^c + \Delta t\boldsymbol{Q}\boldsymbol{\phi} = \boldsymbol{0}, \tag{95}$$

$$\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{u}^c = \boldsymbol{Q}^{\mathrm{T}}\boldsymbol{u}^*, \tag{96}$$

The system (95) and (96) for $(\boldsymbol{u}^c, \boldsymbol{\phi})$ corresponds to the system (80) and (81). Eliminating $\boldsymbol{u}^c$ gives

$$\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{N}^{-1}\boldsymbol{Q}\boldsymbol{\phi} = -\frac{1}{\Delta t}\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{u}^*. \tag{97}$$

We shall call this equation the *Schur complement pressure equation* [84].

Solving (97) requires inverting $\boldsymbol{N}$, which is not an option since $\boldsymbol{N}^{-1}$ is dense and $\boldsymbol{N}$ is sparse. Several rough approximations $\widetilde{\boldsymbol{N}}^{-1}$ to $\boldsymbol{N}^{-1}$ have therefore been proposed. In other words, we solve

$$\boldsymbol{Q}^{\mathrm{T}}\widetilde{\boldsymbol{N}}^{-1}\boldsymbol{Q}\boldsymbol{\phi} = -\frac{1}{\Delta t}\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{u}^*. \tag{98}$$

The simplest approach is to let $\boldsymbol{N}$ be an approximation to $\boldsymbol{M}$ only, i.e., $\widetilde{\boldsymbol{N}} = \boldsymbol{I}$ in finite difference methods and $\widetilde{\boldsymbol{N}}$ equal to the lumped mass matrix $\boldsymbol{M}$ in finite element methods. The approximation $\widetilde{\boldsymbol{N}} = \boldsymbol{I}$ leaves us with a standard 5- or 7-star Poisson equation. With $\theta = 0$ we recover the simple explicit scheme from the end of Section 5.6, whereas $\theta = 1$ gives an implicit backward scheme of the same nature as the one described in Section 5.2.

To summarize the algorithm at a time level, we first make a prediction $\boldsymbol{u}^*$ from (94), then solve (98) for the pressure increment $\boldsymbol{\phi}$, and then update the velocity and pressure by

$$\boldsymbol{u}^{\ell+1} = \boldsymbol{u}^* - \Delta t\boldsymbol{N}^{-1}\boldsymbol{Q}\boldsymbol{\phi}, \quad \boldsymbol{p}^{\ell+1} = \beta\boldsymbol{p}^\ell + \boldsymbol{\phi}. \tag{99}$$

Let us now comment upon classical numerical methods for the Navier–Stokes equations and show how they can

be considered as special cases of the algorithm in the previous paragraph. The history of operator splitting methods starts in the mid and late 1960s. Harlow and Welch [31] suggested an algorithm which corresponds to $\beta = 0$ in our set up and centered finite differences on a staggered spatial grid. The Poisson equation for $\boldsymbol{\phi}$ hence becomes an equation for $p^{\ell+1}$ directly. Chorin [14] defined a similar method, still with $\beta = 0$, but using a non-staggered grid. Temam [77] developed more or less the same method independently, but with explicit time stepping. Hirt and Cook [32] introduced $\beta = 1$ in our terminology. All of these early contributions started with spatially discrete equations and performed the splitting afterwards. The widely used SIMPLE method [58] consists of choosing $\theta \neq 0$

$$\widetilde{\boldsymbol{N}} = \mathrm{diag}(\boldsymbol{N}) = \mathrm{diag}(\boldsymbol{M} + \theta \Delta t \boldsymbol{R}(\boldsymbol{u}^{\ell})),$$

when solving (98). A method very closely related to SIMPLE is the segregated finite element approach, see e.g. Chapter 7.3 in [35].

Most later developments follow either the approach for the current subsection or the alternative view from Sections 5.1 and 5.2. Much of the focus in the history of operator splitting methods has been on constructing second- and higher-order splittings in the framework of Sections 5.1–5.3, and 5.5, see e.g. [2,4,10].

We remark that the step for $\boldsymbol{u}^*$ is unnecessary if we solve the system (95) and (96) for $(\boldsymbol{u}^c, \boldsymbol{\phi})$ correctly, basically we then solve the "exact" system (89) and (90). The point is that we solve (95) and (96) approximately because we replace $\boldsymbol{N}$ in (95) by $\widetilde{\boldsymbol{N}}$ when we eliminate $\boldsymbol{u}^c$ to form the pressure Eq. (98). The next section presents a framework where the classical methods from the current section appear as one iteration in an iterative solution procedure for the fully implicit system (89) and (90).

5.8. Fully implicit methods

Rannacher [63] and Turek [84] propose a general framework to analyze the efficiency and robustness of operator splitting methods. We first consider the *fully implicit* system (89) and (90). Eliminating $\boldsymbol{u}^{\ell+1}$ yields (cf. the similar elimination for the Stokes problem in Section 3.2)

$$\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{N}^{-1}\boldsymbol{Q}\boldsymbol{p}^{\ell+1} = \frac{1}{\Delta t}\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{N}^{-1}\boldsymbol{q}, \tag{100}$$

which we can call the *Schur complement pressure equation* for the implicit system. Notice that we obtain the same solution for the pressure in both (100) and (89) and (90). The velocity needs to be computed, after $\boldsymbol{p}^{\ell+1}$ is from (89), which requires an efficient solution of linear systems with $\boldsymbol{N}$ as coefficient matrix (multigrid is an option here).

Turek [84] suggests that many common solution strategies can be viewed as special cases of a preconditioned Richardson iteration applied to the Schur complement pressure equation (100). Given a linear system

$$\boldsymbol{B}\boldsymbol{p}^{\ell+1} = \boldsymbol{b},$$

the preconditioned Richardson iteration reads

$$\boldsymbol{p}^{\ell+1,k+1} = \boldsymbol{p}^{\ell+1,k} - \boldsymbol{C}^{-1}(\boldsymbol{B}\boldsymbol{p}^{\ell+1,k} - \boldsymbol{b}), \tag{101}$$

where $\boldsymbol{C}^{-1}$ is a preconditioner and k an iteration counter. The iteration at a time level is started with the pressure solution at the previous time level:

$$\boldsymbol{p}^{\ell+1,0} = \boldsymbol{p}^{\ell}.$$

Applying this approach to the Schur complement pressure equation (100) gives the recursion

$$\boldsymbol{p}^{\ell+1,k+1} = \boldsymbol{p}^{\ell+1,k} - \boldsymbol{C}^{-1}(\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{N}^{-1}\boldsymbol{Q}\boldsymbol{p}^{\ell+1,k} - \frac{1}{\Delta t}\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{N}^{-1}\boldsymbol{q}). \tag{102}$$

We now show that the operator splitting methods from Section 5.7, based on solving $\boldsymbol{u}^*$ from (94), solving (98) for $\boldsymbol{\phi}$, and then updating the velocity and pressure with (99), can be rewritten in the form (102). This allows us to interpret the methods from Section 5.7 in a more general framework and to improve the numerics and generate new schemes.

To show this equivalence, we start with the pressure update (99) and insert (98) and (94) subsequently:

$$\boldsymbol{p}^{\ell+1} = \boldsymbol{p}^{\ell} + \boldsymbol{\phi} \tag{103}$$

$$= \boldsymbol{p}^{\ell} + (\boldsymbol{Q}^{\mathrm{T}}\widetilde{\boldsymbol{N}}^{-1}\boldsymbol{Q})^{-1}\frac{1}{\Delta t}\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{u}^* \tag{104}$$

$$= \boldsymbol{p}^{\ell} + (\boldsymbol{Q}^{\mathrm{T}}\widetilde{\boldsymbol{N}}^{-1}\boldsymbol{Q})^{-1}\frac{1}{\Delta t}\boldsymbol{Q}^{\mathrm{T}} \times (\boldsymbol{N}^{-1}\boldsymbol{q} - \Delta t\boldsymbol{N}^{-1}\boldsymbol{Q}\boldsymbol{p}^{\ell}), \tag{105}$$

$$= \boldsymbol{p}^{\ell} - (\boldsymbol{Q}^{\mathrm{T}}\widetilde{\boldsymbol{N}}^{-1}\boldsymbol{Q})^{-1} \times \left(\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{N}^{-1}\boldsymbol{Q}\boldsymbol{p}^{\ell} - \frac{1}{\Delta t}\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{N}^{-1}\boldsymbol{q}\right). \tag{106}$$

We have assumed that $\beta = 1$, since this is the case in the Richardson iteration. Eq. (106) can be generalized to an iteration on $\boldsymbol{p}^{\ell+1}$:

$$\boldsymbol{p}^{\ell+1,k+1} = \boldsymbol{p}^{\ell+1,k} - (\boldsymbol{Q}^{\mathrm{T}}\widetilde{\boldsymbol{N}}^{-1}\boldsymbol{Q})^{-1}\left(\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{N}^{-1}\boldsymbol{Q}\boldsymbol{p}^{\ell+1,k} - \frac{1}{\Delta t}\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{N}^{-1}\boldsymbol{q}\right). \tag{107}$$

We see that (107) is consistent with (106) for the first iteration $k = 1$ if $\boldsymbol{p}^{\ell+1,0} = \boldsymbol{p}^{\ell}$. Moreover, we notice that (107) is identical to (102), *provided we choose the preconditioner* $\boldsymbol{C}$ as $\boldsymbol{Q}^{\mathrm{T}}\widetilde{\boldsymbol{N}}\boldsymbol{Q}$. In other words, classical operator splitting methods from Section 5.7 can be viewed as one preconditioned Richardson iteration on the fully implicit system (89) and (90) (though formulated as (100)). If we perform more iterations in (107), we

essentially have an Uzawa algorithm for the original fully implicit system (89) and (90). Using more than one iteration corresponds to iterating on the pressure in (94), i.e., we solve (94) and (98) more than once at each time level, using the most recent pressure approximation to $\boldsymbol{p}^{\ell+1}$ in (94).

The various choices of $\widetilde{\boldsymbol{N}}$ outlined in Section 5.7 resemble various classical operator splitting methods when used in the preconditioner $\boldsymbol{C} = \boldsymbol{Q}^{\mathrm{T}}\widetilde{\boldsymbol{N}}\boldsymbol{Q}$ in the framework (107). This includes explicit and implicit projection methods, pressure correction methods, SIMPLE variants, and also Uzawa methods and the Vanka smoother used in multigrid methods [83].

With the classical operator splitting methods reformulated as one iteration of an iterative method for the original fully implicit system, one can more easily attack the fully implicit system directly; the building blocks needed are fast solvers for $\boldsymbol{N}^{-1}$ and $(\boldsymbol{Q}^{\mathrm{T}}\widetilde{\boldsymbol{N}}\boldsymbol{Q})^{-1}$, but these are already available in software for the classical methods. In this set up, solving the fully implicit mixed system (89) and (90) is from an implementational point of view not more complicated than using a classical operator splitting method from Section 5.7 method repeatedly. This is worth noticing because people tend to implement and use the classical methods because of their numerical simplicity compared with the fully implicit mixed system.

Noticing that (107) is an Uzawa method, we could introduce inexact Uzawa methods [18], where $\boldsymbol{N}^{-1}$ is replaced by $\widetilde{\boldsymbol{N}}^{-1}$ in the right-hand side of (107) (with some additional terms for making (107) consistent with the original linear system). This can represent significant computational savings. One view of such an approach is that one speeds up solving the predictor step (94) when we iterate over the predictor–correction equations in the classical methods.

The system (82) and (83) can easily be augmented with stabilitzation terms, resulting in a modification of the system (89) and (90):

$$\boldsymbol{N}\boldsymbol{u}^{\ell+1} + \Delta t\boldsymbol{Q}\boldsymbol{p}^{\ell+1} = \boldsymbol{q}, \tag{108}$$

$$\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{u}^{\ell+1} - \epsilon\boldsymbol{D}\boldsymbol{p}^{\ell+1} = -\epsilon\boldsymbol{d}. \tag{109}$$

Eliminating $\boldsymbol{u}^{\ell+1}$ yields

$$(\Delta t\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{N}^{-1}\boldsymbol{Q} + \epsilon\boldsymbol{D})\boldsymbol{p}^{\ell+1} = \boldsymbol{Q}^{\mathrm{T}}\boldsymbol{N}^{-1}\boldsymbol{q} + \epsilon\boldsymbol{d}. \tag{110}$$

With this stabilization one can avoid mixed finite elements or staggered finite difference/volume grids.

Let us now discuss how the preconditioner $\boldsymbol{C}$ can be chosen more generally. If we define the error in iteration k as $\boldsymbol{e}^k = p^{\ell+1,k} - p^{\ell+1}$, one can easily show that $\boldsymbol{e}^k = (\boldsymbol{I} - \boldsymbol{C}\boldsymbol{B})\boldsymbol{e}^{k-1}$. One central question is if just one iteration is enough and under what conditions the iteration is convergent. The latter property is fulfilled if the modulus of the eigenvalues of the amplification matrix $\boldsymbol{I} - \boldsymbol{C}\boldsymbol{B}$ are less than unity. Hence, the choice of $\boldsymbol{C}$ is important both with respect to efficiency and robustness of the solution method.

Turek proposes an efficient preconditioner C^{-1} for (101):

$$\boldsymbol{C}^{-1} = \alpha_{\mathrm{R}}\boldsymbol{B}_{\mathrm{R}}^{-1} + \alpha_{\mathrm{D}}\boldsymbol{B}_{\mathrm{D}}^{-1} + \alpha_{\mathrm{K}}\boldsymbol{B}_{\mathrm{K}}^{-1}, \tag{111}$$

where

- $\boldsymbol{B}_{\mathrm{R}}$ is an optimal (reactive) preconditioner for $\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{Q}$,
- $\boldsymbol{B}_{\mathrm{D}}$ is an optimal (diffusive) preconditioner for $\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{Q}$,
- $\boldsymbol{B}_{\mathrm{K}}$ is an optimal (convective) preconditioner for $\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{Q}$.

Both $\boldsymbol{B}_{\mathrm{R}}$ and $\boldsymbol{B}_{\mathrm{D}}$ can be constructed optimally by standard methods; $\boldsymbol{B}_{\mathrm{R}}$ can be constructed by a multigrid sweep on a Poisson-type equation, and $\boldsymbol{B}_{\mathrm{D}}$ can be made simply by an inversion of a lumped mass matrix. However, no optimal preconditioner is known for $\boldsymbol{B}_{\mathrm{K}}$. It is assumed that $\epsilon\boldsymbol{D}$ does not change the condition number significantly and it is therefore not considered in the preconditioner.

It is also possible to improve the convergence and in particular the robustness by utilizing other iterative methods than the Richardson iteration, which is the simplest iterative method of all. One particular attractive class of methods is the Krylov (or conjugate gradient-like) methods. Methods like GMRES are in principle always convergent, but the convergence is highly dependent on the condition number of $\boldsymbol{C}\boldsymbol{B}$. The authors are pursuing these matters for future research. If approximate methods (multigrid or Krylov solvers) are used for $\boldsymbol{N}^{-1}$ too, we have a nested iteration, and for the outer Krylov method to behave as efficiently as expected, it is necessary to solve the inner iterations accurately. Inexact Uzawa methods would hence be attractive since they only involve $\widetilde{\boldsymbol{N}}^{-1}$ in the inner iteration.

There is also a link between operator splitting methods and fully implicit methods without going through the Schur complement pressure equation. In [70] they considered preconditioners for the Stokes problem of the form

$$\begin{bmatrix} \boldsymbol{N} & \boldsymbol{Q} \\ \boldsymbol{Q}^{\mathrm{T}} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{p} \end{bmatrix} = \begin{bmatrix} \boldsymbol{q} \\ \boldsymbol{0} \end{bmatrix} \tag{112}$$

and this lead to preconditioners like

$$\begin{bmatrix} \boldsymbol{C}_1 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{C}_2 \end{bmatrix}, \tag{113}$$

where $\boldsymbol{C}_1$ should be an approximation of the inverse of $\boldsymbol{N}$ and $\boldsymbol{C}_2$ of $\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{N}^{-1}\boldsymbol{Q}$. This "operator splitting" preconditioner was proved to be optimal provided that $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$ were optimal preconditioners for $\boldsymbol{N}$ and $\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{N}^{-1}\boldsymbol{Q}$, respectively. This preconditioner has been extended to the time-dependent fully coupled Stokes

problem in [7] and a preconditioner similar to the one proposed by Turek in (111) is considered in [56].

6. Parallel computing issues

Laminar flow computations in simple geometries, involving a few thousand unknowns, can now be carried out routinely on PCs or workstations. Nevertheless, more demanding flows met in industrial and scientific applications easily require a grid resolution corresponding to 10^5–10^9 unknowns and hence the use of parallel computers. The suitability of the methods for solving the Navier–Stokes equations on parallel computers is therefore of importance.

The operator splitting methods from Section 5 reduce the Navier–Stokes equations to a sequence of simpler problems. For example, the explicit scheme from Section 5.1 involves explicit updates, in the form of matrix-vector products and vector additions, and the solution of a Poisson equation. Vector additions are trivial to parallelize, and the parallel treatment of products of sparse matrices and vectors is well known. Several methods have been shown to be successful for parallel solution of the Poisson equation, but multigrid seems to be particularly attractive as the method is the most effecient solution approach for the Poisson equation on sequential computers and its concurrent versions are well developed and can be realized with close to optimal speed-up. Using multigrid as a preconditioner for a conjugate gradient method does not alter this picture, since the outer conjugate gradient method just involves inner products of vectors, vector additions, and matrix-vector products. In conclusion, the classical explicit operator splitting methods are very well suited for parallel computing.

The implicit operator splitting methods require, in addition to explicit updates and the solution of a Poisson equation, also the solution of a convection–diffusion equation. Domain decomposition methods at the partial differential equation level can split the global convection–diffusion equation into a series of smaller convection–diffusion problems on subdomains, which can be solved in parallel. The efficiency of this method depends on the convergence rate of the iteration, which shows dependence on the nature of the convection. To achieve sufficient efficiency of the iteration, the domain decomposition approach must be combined with a coarse grid correction [75]. Alternatively, the convection–diffusion equation can be viewed as a linear system, with non-symmetric coefficient matrix, to be solved in parallel. Using a conjugate gradient-like method, such as GMRES og BiCGStab, combined with multigrid as preconditioner, yields a solution method whose parallel version is well established and can be realized with close to optimal speed-up. Other popular preconditioners may be more challenging to parallelize well; incomplete factorizations fall in this category. The overall performance of the parallel convection–diffusion solver depends on choices of numerical degrees of freedom in the linear solver, but these difficulties are present also in the sequential version of the method. To summarize, the implicit operator splitting methods are well suited for parallel computers if a good multigrid or domain decomposition preconditioner can be found for the corresponding sequential problem.

The classical fully implicit method for the Navier–Stokes equations normally applies variants of Gaussian elimiation as linear solver (after a linearization of the nonlinear system of algebraic equations by, e.g., some Newton-like method). Parallel versions of sparse and banded Gaussian elimination procedures are being developed, but such methods are much more difficult to implement efficiently on parallel computers than the iterative solvers discussed above.

Solving the fully implicit system for the Navier–Stokes equations by iterative strategies (again after a linearization of the nonlinear system), basically means running an iterative method, like Richardson iteration or a Krylov solver, combined with a suitable preconditioner. In the case we choose the preconditioner to be typical steps in operator splitting methods, as described in Section 5.8, fully implicit methods parallelize with the same efficiency as the corresponding implicit operator splitting methods.

The pressure stabilization technique from Section 4.1 is actually a way of formulating the equation of continuity and get rid of the mixed finite element or staggered grid requirement. This approach is typically used in combination with operator splitting or fully implicit methods, and the extra stabilization terms do not change the parallelization of those methods.

Penalty methods lead to a kind of transient, nonlinear equation of elasticity. After discretizing in time and resolving the nonlinearity, we are left with a partial differential equation or linear system of the same nature as the equation of elasticity. The problem, however, is that the Lame constant λ in this equation is large, which makes it hard to construct efficient *iterative* methods. Large-scale computing with penalty methods is relevant only if efficient iterative methods for the sequential problem can be constructed. When these methods are based on domain decomposition and/or involve vector operations, matrix-vector products, and multigrid building blocks, parallelization is feasible, see [68] for a promising approach.

The classical artificial compressibility method from Section 4.3 is a purely explicit method, just containing explicit updates, and is hence trivial to parallelize well. Also when implicit time discretizations are used, we get matrix systems that can, in principle, be solved in a

parallel fashion using the same methods as we described for the implicit operator splitting approach.

7. Software

Development of robust CFD software is a complicated and time consuming task. Most scientists and engineers who need to solve incompressible viscous fluid flow problems must therefore rely on available software packages. An overview of CFD software is available on the Internet [37], and here we shall just mention a few widely distributed packages.

PHOENICS [50], which appeared in 1981, was the first general-purpose tool for CFD and is now probably the most widely used CFD software. The numerical foundation is the finite volume method in the notation of Patankar [58]. The user can add code and solve equations that are not supported by the package. FLUENT [46] is another general-purpose CFD package that addresses laminar and turbulent incompressible flow, also in combination with combustion and multiple phases. The spatial discretization employs a finite volume technique applicable to unstructured meshes. FIDAP [43] is a general-purpose fluid flow package quite similar to FLUENT, but it applies finite elements for the spatial discretization. CFX [39] is another modern, general-purpose package addressing complex flow problems in the process and chemical industries, including turbulence, multi-phase flow, bubble-driven flow, combustion, flames, and so on. Flow-3D [45] offers special techniques for and specializes in incompressible viscous free surface flow, but the package can be used for more standard external and confined flows as well. Another general-purpose CFD package is CFD2000 [38], which uses finite volumes on curvelinear grids and handles incompressible as well as compressible flows, with turbulence, heat transfer, multiple phases, and chemical reactions. CFD++ is a finite volume-based solver with particular emphasis on turbulent flow. It handles many different types of grids and flow regimes (from incompressible to hypersonic). ANSYS/FLOTRAN [44] is a CFD package contained in the ANSYS family of finite element codes. The tight integration with other ANSYS codes for heat transfer, elasticity, and electromagnetism makes it feasible to perform multi-physics simulations, e.g., fluid–structure interactions and micro-electro-mechanical systems (MEMS). The CFD software mentioned so far covers advanced, commercial, general-purpose tools that offer a complete problem solving environment, with user-friendly grid generation and visualization facilities in addition to the numerical engines.

FEATFLOW [42] is a free package implementing the framework from Section 5.8, with finite elements in space, multigrid for solving linear and nonlinear systems, and an emphasis on computational efficiency and robustness. FEMLAB is a package built on top of Matlab and offers easy set-up of incompressible flow problems, also coupled with heat transfer, electromagnetism, and elasticity. Fastflo [41] is a code of a similar nature. Mouse [48] is a modern finite volume-based software library, packed with ready-made CFD programs. Some flexible, general-purpose, *programmable* environments and libraries with important applications to CFD are Diffpack [40], FOAM [47], Overture [49], and UG [51].

The quality and robustness of the numerics even in advanced packages, especially when simulating complex multi-fluid flows, may be questionable as we know that our understanding of how to solve the building block (1) and (2) is still limited. More benchmark problem initiatives [84] are needed to classify flow cases and methods whose numerical results are reliable.

The complexity of CFD applications makes a demand for flexibility, where different splitting approaches, different space and time discretizations, different linear and nonlinear system solvers, and different governing equations can be freely combined. Much of the current CFD software, written as large, stand-alone Fortran 77 programs, lacks this freedom of choice because of an inflexible software design. There is now a growing interest in methodologies for better design of CFD software based on, e.g., object-oriented programming techniques [40,47–49,53,57,81,85].

8. Future directions of research

During the last decades there has been tremendous progress in computational fluid dynamics. Even if some of the most basic tools are well understood by now, there are still challenging problems related to numerical methods for incompressible Navier–Stokes equation. In fact, the simplified Stokes problem (11) and (12), especially when (11) is augmented with a time derivative $\partial \boldsymbol{v}/\partial t$, is not fully understood when it comes to stable discretizations [56], efficient solution of the resulting linear systems (in an implicit formulation), as well as a priori and a posteriori error estimates. It therefore seems fruitful to still address simplified versions of the Navier–Stokes equations to develop and understand numerical methods.

The most popular time stepping methods for the Navier–Stokes equation are not fully implicit. As explained in Section 5.1, an operator splitting strategy can be used to essentially decouple the updating of the velocity and the pressure. However, the choice of boundary conditions for the pressure in these procedures is problematic. The pressure is closely tied to the incompressibility condition for the velocity, and any decoupling therefore has drawbacks. As in many other fields,

engineers prefer to work with robust and stable numerical engines, even at a cost of decreased computational efficiency. This usually implies that fully implicit codes are preferred in industry. Hence, we think that fully implicit schemes, like the ones discussed in Sections 5.2 and 5.8 should be further developed. The study of efficient preconditioners, which are robust with respect to physical and numerical parameters, seems essential. An attractive framework would be to reuse the experience and knowledge built up around (classical) operator splitting schemes as preconditioners for the more robust fully coupled and implicit scheme. For efficiency, multigrid cycles should be used in such preconditioners, but the optimal combination of multigrid building blocks (smoothers, cycling strategies, etc.) are not yet known for fully implicit formulations. In addition, the use of parallel computational techniques, like domain decomposition algorithms, is essential for performing fine scale simulations. As the gap between CPU power and memory access efficiency increases, it might be questionable whether today's numerics are well suited for the coming generation of high-performance computers. This calls for paying closer attention to the interplay between hardware and numerics.

Another topic which will be essential for doing fine scale computations is the development of adaptive algorithms using suitable error estimators. In particular, for simulations in complex geometries such tools will be unavoidable [63,84]. The ultimate goal is to allow the engineer to specify an error tolerance and let adaptive algorithms and error estimators find the corresponding discretization parameters and solution strategies. This will be an important goal because we cannot expect that sufficient numerical and physical competence will be present among all users when CFD becomes a cheap, widely accessible, and apparently simple-to-use tool.

We have already mentioned the particular need for robust Navier–Stokes solvers when such solvers are embedded in complex fluid models, involving multi-species fluids, turbulence, heat transfer, chemical reactions, and coupling to structural deformations. Many of the future scientific water resources applications of incompressible viscous flow will involve such composite models. For example, a better understanding of consolidation of porous media may be based on studying coupled models of Navier–Stokes equations and the equations of elasto-visco-plasticity at the pore level. Geologists are especially interested in cracking mechanisms, where the desire is to simulate deformation, contact, and fracture of sand grains embedded in a viscous fluid. The science of reactive porous media flow has many open questions, and fundamental studies may benefit from simulating pore level multi-phase flow coupled with chemistry models, in particular in the vicinity of rock "walls". Fortunately, the computing technology necessary for realizing such studies is the same that is needed for complex flow problems found in, e.g., the car industry and physiology.

9. Concluding remarks

We have in this paper presented a basic introduction to some well-known and widely used numerical methods for the incompressible Navier–Stokes equations. The emphasis has been on the fundamental underlying ideas for discretizing the system of partial differential equations. By hiding the details of the spatial discretization (finite elements, finite differences, finite volumes, spectral methods), we hope to better explain the close relationship between different numerical schools, in particular finite differences/volumes and finite elements; most of the schools apply the same fundamental reasoning and arrive in most cases at the same algebraic equations (modulo effects from different treatment of boundary conditions). Naturally, our main emphasis is on advancing the equations in time and especially how to split the equations into more tractable systems. The methods covered herein include artificial compressibility, penalty function formulations, fully implicit schemes, and operator splitting techniques (and their aliases projection methods, fractional step methods, pressure correction strategies).

As an extension of the introduction to operator splitting methods we have proposed a framework, inspired by Rannacher [63] and Turek [84], where many classical and popular methods can be viewed as certain preconditioned iterative methods applied to a robust, fully implicit formulation of the Navier–Stokes equations. This opens up the possibility of a more general view of the classical methods and for constructing an endless series of new solution strategies, all of them trying to solve the fully implicit system iteratively. One such strategy pursued by the authors has been outlined.

Due to page limitations, several important topics had to be left out or only briefly commented upon. Such topics include, among others, detailed information about finite element, finite difference, and finite volume discretization techniques, spectral methods, higher-order temporal schemes, "upwind" techniques for high Reynolds number flow, discretizing boundary conditions, least-squares finite element formulations, vorticity-streamfunction formulations, grid generation techniques, as well as adaptivity methods based on error estimation and control. Relevant textbooks for references and further reading about these subjects are [15,20,27,30,61,84,86,89].

The lack of a discussion of special methods for high Reynolds number flow, where a careful treatment of the convective term $\boldsymbol{v} \cdot \nabla \boldsymbol{v}$ is important, may give the impression that these flow regimes are beyond scope of the presented solution methods. This is not true; the same

basic solution strategies can still be applied, but then in combination special spatial discretization techniques for the convective term (typically upwind differencing or Petrov–Galerkin finite element methods). More sophisticated approaches for highly convective flow apply special space-time integrations, and this may affect the operator splitting strategies. Within a fractional step approach (as in Section 5.5), the convection term is isolated in a Burgers equation, such that the treatment of high Reynolds number flow is then limited to an advanced numerical treatment of a Burgers equation.

No numerical experiments have been presented in this paper. Results of extensive numerical investigations with different solution strategies for the Navier–Stokes equations can be found in textbooks, see [27,84] in particular. The virtual album of fluid motion [42,84] contains an exciting collection of animations for numerous flow configurations.

Finally, we mention that incompressible viscous flow can also be computed by means of lattice gas or lattice Boltzman methods [12]. These methods are statistical in nature and involve collisions of a large number of particles moving in a lattice. The approach is particularly attractive for simulation of complex multi-phase flows where the qualitative flow behavior is in focus. When computational accuracy is important, and only one fluid is flowing in the laminar regime, discretization of the Navier–Stokes equations as outlined in the present paper seems to be the most efficient and robust solution strategy.

Acknowledgements

The authors thank Dr. Torgeir Rusten for numerous fruitful discussions and useful input to the present manuscript.

References

[1] Aliabadi S, Tezduyar T. 3D simulation of two-fluid and free-surface flows with the stabilized-finite-element/interface-capturing method. In: Atluri S, O'Donoghue P, editors. Modeling and Simulation Based Engineering; Proceedings of International Conference on Computational Engineering Science, Atlanta, Georgial; 1998.

[2] Almgren AS, Bell JB, Szymczak WG. A numerical method for the incompressible Navier–Stokes equations based on an approximate projection. SIAM J Sci Comput 1996;17(2):358–69.

[3] Anderson JD. Computational fluid dynamics—The basics with applications. New York: McGraw-Hill; 1995.

[4] Bell JB, Colella P, Glaz HM. A second order projection method for the incompressible Navier–Stokes equations. J Comput Phys 1989;85:257–83.

[5] Bercovier M, Pironneau O. Error estimates for finite element method solution of the Stokes problem in the primitive variables. Numer Math 1979:211–24.

[6] Blunt MJ. Effects of heterogeneity and wetting on relative permeability using pore level modeling. SPE J 1997:70–87.

[7] Bramble JH, Pasciak JE. Iterative techniques for time dependent stokes problems. Math Appl 1997.

[8] Brezzi F, Fortin M. Mixed and hybrid finite element methods. Berlin: Springer; 1991.

[9] Brooks AN, Hughes TJR. A streamline upwind/Petrov–Galerkin finite element formulation for advection domainated flows with particular emphasis on the incompressible Navier–Stokes equations. Comput Meth Appl Mech Eng 1982:199–259.

[10] Brown DL, Cortez R, Minion ML. Accurate projection methods for the incompressible Navier–Stokes equations. J Comput Phys 2001:464–99.

[11] Canuto C, Hussaini MY, Quarteroni A, Zang TA. Spectral methods in fluid dynamics. Springer Series in computational physics. Berlin: Springer; 1988.

[12] Chen S, Doolen GD. Lattice Boltzmann method for fluid flows. Ann Rev Fluid Mech 1998:329–64.

[13] Chorin AJ. A numerical method for solving incompressible viscous flow problems. J Comput Phys 1967;2:12–26.

[14] Chorin AJ. Numerical solution of the Navier–Stokes equations. Math Comput 1968;22:745–62.

[15] Chung T. Computational fluid dynamics. Cambridge: Cambridge University Press; 2001.

[16] Dean EJ, Glowinski R. On some finite element methods for the numerical simulation of incompressible viscous flow. In: Gunzburger MD, Nicolaides RA, editors. Incompressible computational fluid dynamics; trends and advances. Cambridge: Cambridge University Press; 1993.

[17] WE, Liu J-G. Finite difference schemes for incompressible flows in the velocity-impulse density formulation. J Comput Phys 1997.

[18] Elman HC, Golub G. Inexact and preconditioned Uzawa algorithms for saddle point problems. SIAM J Numer Anal 1994;31:1645–61.

[19] Engelman MS, Sani RI, Gresho PM, Bercovier M. Consistent vs. reduced integration penalty methods for incompressible media using several old and new elements. Int J Numer Meth Fluids 1982;2:25–42.

[20] Ferziger JH, Perić M. Computational methods for fluid dynamics. Berlin: Springer; 1996.

[21] Fletcher CAJ. Computational techniques for fluid dynamics Vol I and II. Springer Series in computational physics. Berlin: Springer; 1988.

[22] Fortin M, Glowinski R. In: Augmented Lagrangian methods: applications to the numerical solution of boundary-value problems. Series: studies in mathematics and its applications, vol. 15. North-Holland: Amsterdam, The Netherlands; 1983.

[23] Fukumori E, Wake A. The linkage between penalty function method and second viscosity applied to Navier–Stokes equation. Comput Mech 1991.

[24] Gignard S, Grilli ST, Marcer R, Rey V. Computation of shoaling and breaking waves in nearshore areas by the coupling of bem and vof methods. In: Dæhlen M, Tveito A, editors, Proceedings of 9th Offshore and Polar Engineering Conference (ISOPE99), vol. III. 1999. p. 304–9.

[25] Girault V, Raviart PA. Finite element methods for Navier–Stokes equations. Berlin: Springer; 1986.

[26] Glowinski R. Numerical methods for nonlinear variational problems. Berlin: Springer; 1984.

[27] Gresho PM, Sani RL. Incompressible flow and the finite element method. New York: Wiley; 1998.

[28] Griebel M, Dornseifer T, Neunhoeffer T. Numerical simulation in fluid dynamics: a practical introduction, SIAM: Philadelphia, PL; 1997 [Also available in German: Numerische Simulation in der Strömungsmechanik: Eine Praxisorienterte Einfürung, Vieweg Lehrbuch Scientific Computing, 1995].

[29] Gunzburger M. Finite element methods for viscous incompressible flows. New York: Academic Press; 1989.
[30] Gunzburger MD, Nicolaides RA. Incompressible computational fluid dynamics; Trends and advances. Cambridge: Cambridge University Press; 1993.
[31] Harlow FH, Welch JE. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. Phys Fluids 1965;8:2182–9.
[32] Hirt CW, Cook JL. Calculating three-dimensional flows around structures and over rough terrain. J Comput Phys 1972;10:324–40.
[33] Ho LW, Rnquist EM. Spectral element solution of steady incompressible viscous free-surface flows. Finite Elem Anal Des 1994:207–27.
[34] Hughes TJR, Liu WK, Brooks A. Finite element analysis of incompressible viscous flows by the penalty function formulation. J Comput Phys 1979;30:1–60.
[35] Fluent Inc. FIDAP theory manual. See e.g. Available from: http://fortuitous.ncsa.uiuc.edu/fidapdocumentation/.
[36] Internet. CFD Online's book page. Available from: http://www.cfd-online.com/Books/.
[37] Internet. CFD online's software overview. Available from: http://www.cfd-online.com/Resources/soft.html.
[38] Internet. CFD2000 software package. Available from: http://www.adaptive-research.com/.
[39] Internet. CFX software package. Available from: http://www.software.aeat.com/cfx/.
[40] Internet. Diffpack software package. Available from: http://www.diffpack.com.
[41] Internet. Fastflo software package. Available from: http://www.nag.co.uk/simulation/Fastflo/fastflo.html.
[42] Internet. FEATFLOW software package. Available from: http://www.featflow.de.
[43] Internet. FIDAP software package. Available from: http://www.fluent.com/software/fidap/.
[44] Internet. FLOTRAN software package. Available from: http://www.ansys.com/products/flotran.htm.
[45] Internet. Flow-3D software package. Available from: http://www.flow3d.com.
[46] Internet. FLUENT software package. Available from: http://www.fluent.com/software/fluent/.
[47] Internet. FOAM software package. Available from: http://www.nabla.co.uk.
[48] Internet. Mouse software package. Available from: http://fire8.-vug.uni-duisburg.de/MOUSE/.
[49] Internet. Overture software package. Available from: http://www.llnl.gov/casc/Overture/.
[50] Internet. PHOENICS software package. Available from: http://213.210.25.174/phoenics/d_polis/d_info/phover.htm.
[51] Internet. UG software package. Available from: http://cox.i-wr.uni-heidelberg.de/ug/.
[52] Kim J, Moin P. Application of a fractional-step method to incompressible Navier–Stokes equations. J Comput Phys 1985.
[53] Langtangen HP. Computational partial differential equations—Numerical methods and diffpack programming. Lecture notes in computational science and engineering. Berlin: Springer; 1999.
[54] Löhner R. Design of incompressible flow solvers: practical aspects. In: Gunzburger MD, Nicolaides RA, editors. Incompressible computational fluid dynamics; Trends and advances. Cambridge: Cambridge University Press; 1993.
[55] Mardal K-A, Tai X-C, Winther R. Robust finite elements for Darcy–Stokes flow. SIAM J Numer Anal (in press).
[56] Mardal K-A, Winther R. Uniform preconditioners for the time dependent Stokes problem. 2002 [in preparation].
[57] Munthe O, Langtangen HP. Finite elements and object-oriented implementation techniques in computational fluid dynamics. Comput Meth Appl Mech Eng 2000;190:865–88.
[58] Patankar SV. Numerical heat transfer and fluid flow. Series in computational methods in mechanics and thermal sciences. New York: McGraw-Hill; 1980.
[59] Peyret R, Taylor TD. Computational methods for fluid flow. Springer series in computational physics. Berlin: Springer; 1983.
[60] Pironneau O. The finite element methods for fluids. New York: John Wiley and Sons; 1989.
[61] Quarteroni A, Valli A. Numerical approximation of partial differential equations. Springer series in computational mathematics. Berlin: Springer; 1994.
[62] R. Rannacher. On Chorin's projection method for incompressible Navier–Stokes equations. In: Heywood JG, Masuda K,Rautmann R, Solinikov VA, editors. Theory of the Navier–Stokes equations. Advances in Mathematics for Applied Sciences. Springer; 1998. Based on an Oberwolfach Conference on the Navier–Stokes Equations.
[63] R. Rannacher, Finite element methods for the incompressible Navier–Stokes equation. 1999 Available from: http://www.iwr.u-ni-heidelberg.de/sfb359/Preprints1999.html.
[64] Reddy JN. On penalty function methods in the finite element analysis of flow problems. Int J Numer Meth Fluids 1982;18:853–70.
[65] Reddy JN. An introduction to the finite element method. New York: McGraw-Hill; 1993.
[66] Reddy JN, Gartling DK. The finite element method in heat transfer and fluid dynamics. Boca Raton: CRC Press; 1994.
[67] Reddy MP, Reddy JN. Multigrid methods to accelerate the convergence of element-by-element solution algorithms for viscous incompressible flows. Comput Meth Appl Mech Eng 1996;132:179–93.
[68] Reddy MP, Reddy JN, Akay HU. Penalty finite element analysis of incompressible flows using element-by-element solution algorithms. Comput Meth Appl Mech Eng 1992;100:169–205.
[69] Rudman M. A volume-tracking method for incompressible multifluid flows with large density variations. Int J Numer Meth Fluids 1998:357–78.
[70] Rusten T, Winther R. A preconditioned iterative method for saddlepoint problems. SIAM J Matrix Anal 1992.
[71] Sani RL, Gresho PM, Lee RL, Griffiths DF. The cause and cure (?) of the spurious pressures generated by certain FEM solutions of the incompressible Navier–Stokes equations: part 1 (part 2). Int J Numer Meth Fluids 1981;1:17–43.
[72] Scardovelli R, Zaleski S. Direct numerical simulation of free-surface and interfacial flow. Ann Rev Fluid Mech 1999:567–603.
[73] Shen J. On error estimates of the projection methods for the Navier–Stokes equations: First-order schemes. SIAM J Numer Anal 1996.
[74] Shen J. On error estimates of the projection methods for the Navier–Stokes equations: Second-order schemes. Math Comput 1996.
[75] Smith B, Bjørstad P, Gropp W. Domain decomposition—Parallel multilevel methods for elliptic partial differential equations. Cambridge: Cambridge University Press; 1996.
[76] Taylor C, Hood P. A numerical solution of the Navier–Stokes equations using the finite element method. J Comput Phys 1973;1:73–100.
[77] Temam R. Sur l'approximation de la solution des équations de Navier–Stokes par la méthode des pas fractionnaires. Arc Ration Mech Anal 1969;32:377–85.
[78] Tezduyar TE. Adaptive determination of the finite element stabilization parameters. In: Computational Fluid Dynamics Conference Proceeding. 2001.
[79] Tezduyar TE, Aliabadi S, Behr M. Interface-capturing technique (EDICT) for computation of unsteady flows with interfaces. Comput Meth Appl Mech Eng 1998;155:235–48.
[80] Thompson KE, Fogler HS. Modeling flow in disordered packed beds from pore-scale fluid mechanics. AIChE J 1997;43(6):1377.

[81] Thuné M, Mossberg E, Olsson P, Rantakokko J, Åhlander K, Otto K. Object-oriented construction of parallel PDE solvers. In: Arge E, Bruaset AM, Langtangen HP, editors. Modern Software Tools for Scientific Computing. Basel: Birkhäuser; 1997. p. 203–26.
[82] Tobiska L, Verfürth R. Analysis of a streamline diffusion finite element method for the Stokes and Navier–Stokes equations. SIAM J Numer Anal 1996.
[83] Turek S. A comparative study of time-stepping techniques for the incompressible Navier–Stokes equations: From fully implicit non-linear schemes to semi-implicit projection methods. Int J Numer Meth Fluids 1996;22:987–1011.
[84] Turek S. Efficient solvers for incompressible flow problems. Berlin: Springer; 1999.
[85] Weller HG, Tabor G, Jasak H, Fureby C. A tensorial approach to computational continuum mechanics using object orientated techniques. Comput Phys 1998;12(6):620–31.
[86] Wesseling P. Principles of computational fluid dynamics. Springer series in computational mathematics. Berlin: Springer; 2001.
[87] Wille SØ. Nodal operator splitting adaptive finite element algorithms for the Navier–Stokes equations. Int J Numer Meth Fluids 1998:959–75.
[88] Zienkiewicz OC, Morgan K. Finite elements and approximation. New York: Wiley; 1983.
[89] Zienkiewicz OC, Taylor RL. The finite element method [3 volumes]. 5th ed. New York: McGraw-Hill; 2000.